UNIVERSIDAD CARLOS III DE MADRID
Departamento de Ingeniería Telemática

PROYECTO FIN DE CARRERA

DISEÑO Y DESARROLLO DE UN MARCO DE PRUEBAS Y DETECCIÓN DE NATS

Autor: Francisco José Blázquez Sánchez

Tutor: Francisco Valera Pintor

Leganés, Octubre de 2009

Título: DISEÑO Y DESARROLLO DE UN MARCO DE PRUEBAS Y DETECCIÓN DE NATS
Autor: Francisco José Blázquez Sánchez
Director: Francisco Valera Pintor

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 22 de Octubre de 2009 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A Alba, a mi familia, a Eva, a mis compañeros de Oberon Sistemas, Javier y Tino y a mi tutor Francisco Valera, gracias.

Resumen

"El funcionamiento de NAT varia de una implementación a otra. Es por lo tanto muy difícil para las aplicaciones finales predecir o descubrir el comportamiento de estos dispositivos. La predicción y el descubrimiento del funcionamiento del NAT es importante para diseñar protocolos de aplicación y técnicas de traspaso de NAT que trabajen de manera fiable en las redes existentes. Esta situación es especialmente problemática para aplicaciones extremo a extremo donde uno o ambos extremos se encuentran detrás de un NAT, como son los juegos multiusuario, aplicaciones multimedia interactivas o aplicaciones P2P."

Descripción del grupo de trabajo behave del IETF.

He seleccionado como resumen del trabajo este extracto de la página Web del IETF por dos motivos. Primero, porque el desarrollo se ha basado completamente en los estudios realizados y publicados por este grupo de trabajo. Después porque en este párrafo encontramos el motivo del desarrollo del proyecto que es el de la predicción y el descubrimiento del comportamiento del NAT y la problemática que suponen este tipo de dispositivos en los nuevos usos de Internet.

Como resultado del proyecto obtendremos una plataforma independiente de simulación de comunicaciones UDP a través de NAT, que permita el desarrollo de pruebas de detección y descubrimiento de NAT mediante el envío de mensajes STUN. Para ello desarrollaremos fundamentalmente tres componentes software que representen los tres elementos necesarios en este escenario, los extremos finales (el cliente y el servidor STUN) y obviamente la caja NAT. El NAT desarrollado simulará diferentes comportamientos configurables de forma que se abarque el mayor número de situaciones en las que se pueden encontrar los extremos de la comunicación.

Adicionalmente, se realizará pruebas de detección en escenarios reales utilizando el cliente STUN y los procesos de detección desarrollados para el entorno de simulación.

Palabras clave: NAT, STUN, UDP, IETF, behave, pcap, sockets raw.

Abstract

"The behavior of NATs varies from one implementation to another. As a result it is very difficult for applications to predict or discover the behavior of these devices. Predicting and/or discovering the behavior of NATs is important for designing application protocols and NAT traversal techniques that work reliably in existing networks. This situation is especially problematic for end-to-end applications where one or both end-points are behind a NAT, such as multiuser games, interactive multimedia and P2P download."

Description of working group behave of IETF

I have selected as summary of the work this extract of the web page of the IETF for two motives. First, because the development has been based completely on the studies realized and published by this group of work. Later because in this paragraph we find the motive of the development of the project that is that of the prediction and the discovery of the behavior of the NAT and the problematics that this type of devices suppose in the new uses of Internet.

As result of the project we will obtain a platform independent for simulation of communications UDP across NAT, which allows the development of tests of detection and NATs discovery using the sending of messages STUN. For it we will develop fundamentally three software components that represent the three necessary elements in this scene, the final ends (the client and the server STUN) and obviously the box NAT. The developed NAT will simulate different behaviors configurables so that the major number of situations is included in those that can be the ends of the communication.

Additional, tests of detection will be realized in real scenes using the client STUN and the processes of detection developed for the simulation environment.

Keywords: NAT, STUN, UDP, IETF, behave, pcap, sockets raw.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

IPv4 usa direcciones de 32 bits, limitándolas a 2^{32} (4.294.967.296) direcciones únicas. Por el enorme crecimiento que ha tenido Internet, combinado con el hecho de que haya desperdicio de direcciones en muchos casos, ya hace varios años que se vio que **escaseaban las direcciones IPv4**. Esta limitación impulsó **IPv6**, que se espera que termine reemplazando a IPv4.

Para lidiar momentáneamente con este problema de escasez de direcciones se empezó a emplear la técnica NAT.

NAT (Network Address Translation - Traducción de Dirección de Red), es un método por el cual las direcciones IP son mapeadas desde un espacio de direcciones a otro, proporcionando enrutamiento transparente en los extremos finales, de forma que al traducir las direcciones se multiplican las posibilidades efectivas de asignación (IP Network Address Translator (NAT) Terminology and Considerations [RFC 2663]).

Una pasarela NAT cambia la dirección origen en cada paquete de salida y, dependiendo del método, también el puerto origen para que sea único. Estas traducciones de dirección se almacenan en una tabla, para recordar qué dirección y puerto le

corresponde a cada dispositivo cliente y así saber donde deben regresar los paquetes de respuesta. Si un paquete que intenta ingresar a la red interna no existe en la tabla de traducciones, entonces es descartado. Debido a este comportamiento, se puede definir en la tabla que en un determinado puerto y dirección se pueda acceder a un determinado dispositivo, como por ejemplo un servidor web, lo que se denomina NAT inverso o DNAT (Destination NAT).

Los **ISP** también pueden utilizar NAT para aliviar la escasez de direcciones IP para los usuarios de cable y **ADSL**. En este caso el ISP le asigna una dirección a cada usuario. Cuando los paquetes de las máquinas de usuario salen del ISP atraviesan una caja NAT que los traduce a la verdadera dirección de Internet del ISP. En el camino de regreso, los paquetes sufren la conversión inversa. En este caso, para el resto de Internet, el ISP y sus usuarios caseros de cable y ADSL se comportan como una compañía grande.

Estos usos de NAT rompen el principio de diseño 'extremo a extremo' ('end-to-end principle'). Este principio de diseño establece que 'es deseable que la operación de los protocolos de comunicación opere en las puntas terminales de la red' y 'ésta – como componente intermedio – no debe tratar de ofrecer ayuda a la aplicación, pues al hacerlo sólo generaría redundancias'. En general la técnica NAT causa muchos problemas con protocolos que transportan direcciones IP en el cuerpo del mensaje. Estos protocolos fallan porque no tienen en cuenta que las direcciones IP que trasportan en el cuerpo de sus mensajes pueden ser en realidad direcciones IP inalcanzables por encontrarse detrás de un NAT y por lo tanto pertenecen al espacio de direcciones IP privadas de una red local. En la [RFC 2993] encontramos la siguiente reflexión:

'Publicado en Mayo de 1994, escrito por K. Egevang and P. Francis, [RFC 1631] se definió NAT como una forma de aliviar el índice de crecimiento del uso de direcciones IPv4. Pero los autores se mostraron preocupados acerca del impacto de esta tecnología. En su documento, apuntaron la necesidad de experimentar y ver que aplicaciones podrían verse afectadas por la manipulación de las cabeceras de NAT, incluso antes de que existiera alguna experiencia operacional significativa. Esto es aún mas evidente en sus conclusiones: 'NAT tiene varias características negativas que le convierten en una solución inapropiada a largo plazo y pueden hacerla inapropiada incluso como solución a corto plazo.'

Hoy en día, seis años después y en contra de las predicciones, el uso de NAT se ha extendido en Internet. Muchos proclaman que NAT es la solución a la disponibilidad de direcciones de Internet y cuestionan la necesidad de continuar con el desarrollo de IPv6. Reclaman que NAT realiza su función sin efectos serios excepto para unas pocas aplicaciones. Al mismo tiempo otras personas ven miríadas de dificultades originadas por el creciente uso de NAT.'

La técnica NAT implica que bajo determinadas situaciones de tráfico, la asignación de direcciones IP externas a cada usuario de la red privada puede derivar en un **comportamiento no determinístico del NAT**.

Estas características, junto con la falta de acuerdos entre los distintos fabricantes de NAT en cuanto a su funcionamiento, provocan que las aplicaciones finales tengan que lidiar con situaciones muchas veces desconocidas a priori. Por este motivo, el IETF, concretamente el grupo de trabajo **behave** (Behavior Engineering for Hindrance

Avoidance) del área de transporte, ha decidido intentar homogeneizar los distintos tipos de NAT.

Para intentar predecir el comportamiento del NAT bajo situaciones específicas de tráfico, apareció el protocolo STUN.

STUN (Simple Traversal Underneath Network Address Translators) proporciona funciones que permiten a entidades situadas detrás de un NAT, conocer la dirección asignada por el NAT.

STUN es utilizado en la solución ICE proporcionada por el grupo de trabajo MMUSIC (Multiparty Multimedia Session Control) del IETF, al problema del inicio de sesión del protocolo SIP (Session Initiation Protocol). Esta solución queda fuera del alcance del proyecto.

La naturaleza del problema de la observación del estado del NAT es impredecible, en cuanto que la propia utilización de STUN genera tráfico que potencialmente modifica el estado del NAT. En cualquier caso es una técnica muy útil para realizar comprobaciones o en situaciones de carga conocidas a priori.

1.2 Objetivos

El objetivo fundamental del proyecto es el diseño y desarrollo de una herramienta completa de estudio de NAT que permita generar escenarios reales así como evaluar el comportamiento de los diferentes tipos de NAT existentes.

En base a ese objetivo principal, se proponen los siguientes objetivos parciales:

- Diseñar y desarrollar una herramienta de generación de test basada en STUN para la observación del comportamiento de los diferentes tipos de NAT disponibles en la actualidad de acuerdo a la tipología de NAT establecida por el IETF y que podemos encontrar en http://tools.ietf.org/id/draft-ietf-behave-nat-behavior-discovery-07.txt. Esta herramienta deberá ser portable a entornos Windows y Unix.
- Debido a la imposibilidad de contar con un abanico de tipos de NAT suficientemente representativo, se marca como siguiente objetivo el diseño y la implementación de una herramienta de simulación de escenarios típicos de explotación de la técnica NAT, en la que podremos demostrar como la herramienta de generación de test es útil y funcional.

1.3 Fases del desarrollo

El proyecto se completa con las siguientes fases:

- FASE 1: estudio de los diferentes tipos de NAT y de la problemática asociada a la utilización de estos en las comunicaciones extremo a extremo sobre protocolo UDP.
- FASE 2: diseño e implementación de los elementos necesarios del protocolo STUN para la detección de NAT.
- FASE 3: diseño e implementación de las herramientas software necesarias para la creación de pruebas de detección basadas en STUN.
- FASE 4: creación y pruebas de detección básica sobre la herramienta desarrollada.
- FASE 5: diseño e implementación de una herramienta de simulación de escenarios para la observación detallada del comportamiento de NAT ante los test desarrollados en la fase 4.

1.4 Medios empleados

Para el desarrollo del proyecto se ha contado con los siguientes elementos hardware:

- MacBook Pro 2.2 GHz Intel Core 2 Duo 2 GB 667 MHz DDR2
- Router Zyxel de Telefónica.
- Disco duro externo de 100 GB de capacidad.

En cuanto al software, ha sido necesario:

- Windows 2000 Professional
- Windows Server 2003 Enterprise Edition SP 2
- Microsoft Visual Studio .NET 2003
- Mac OS X 10.4.11
- XCode Version 2.5
- Parallels Desktop 3.0 for Mac
- VirtualBox 3.0.6 for Windows hosts
- Microsoft Office 2007
- Microsoft Visio 2007
- WinPCap 4.0.2
- PThreads-Win32 Versión 2.8.0
- Wireshark Versión 1.0.6

1.5 Estructura de la memoria

El documento está estructurado en ocho capítulos.

El primer capítulo, el capítulo actual, presenta una introducción a la problemática del uso del NAT y fija los objetivos del proyecto.

El segundo capítulo describe la técnica NAT, y presenta la clasificación de NAT y el conjunto de requisitos necesarios para conseguir que el mayor número posible de aplicaciones funcione de manera consistente. Esta clasificación y los requisitos han sido extraídos de los trabajos desarrollados por el grupo de trabajo behave del área de transporte del IETF.

El capítulo tercero muestra el resultado del análisis de la solución propuesta para la consecución de los objetivos marcados en el proyecto. Se divide en dos grandes bloques dedicados a STUN y NAT.

El capítulo cuatro desarrolla el diseño de los componentes software necesarios para la implementación de la solución propuesta. En este encontramos las librerías y clases desarrolladas y un diagrama detallado de la solución.

El capítulo cinco muestra en detalle la implementación de las pruebas de detección desarrolladas y presenta la aplicación de simulación creada para la ejecución de estas pruebas. Se describe como instalar la aplicación y la manera de crear los escenarios sobre los que desarrollar las pruebas. También encontramos en este capítulo la información necesaria para interpretar los resultados de la ejecución de las pruebas.

El sexto capítulo contiene el presupuesto final del proyecto y los costes de ejecución del mismo.

El capítulo séptimo expone las conclusiones y posibles trabajos futuros.

El capítulo octavo contiene el glosario de términos usados para la comprensión de documentación.

El capítulo noveno enumera las referencias utilizadas durante el desarrollo del proyecto.

Capítulo 2

NAT

2.1 Introducción

En este capítulo vamos a establecer una clasificación de los diferentes tipos de NAT basada en los estudios realizados por el grupo de trabajo **behave** del **área de transporte** del **IETF**.

El capítulo comienza con una breve definición de NAT y una clasificación inicial y poco rigurosa (extraída de http://es.wikipedia.org/wiki/Network_Address_Translation) de los distintos tipos de NAT.

A continuación se detallan en profundidad los distintos tipos de NAT en función de sus comportamientos ante determinadas situaciones de tráfico De cada uno de los comportamientos observados, el grupo de trabajo behave ha extraído los requerimientos que deben cumplir los NAT para asegurar el correcto funcionamiento del mayor número posible de aplicaciones. Estos requerimientos se detallan tal y como aparecen en Network Address Translation (NAT) Behavioral Requirements for Unicast UDP [RFC 4787]. Algunas de las funcionalidades descritas en este documento explican como se comportan los distintos tipos de NAT en función del modo en que lo observamos, es decir, dependiendo de como realiza la asignación de dirección IP y puerto, la asignación de dirección IP en NAT con más de una dirección pública, la asignación del puerto en

caso de colisión entre dos o más equipos, los conflictos de direcciones entre los espacio de direcciones privado y público, el filtrado de paquetes de entrada…

Finalmente se hace una recopilación de los requerimientos que debe cumplir el NAT ideal y se comentan algunas conclusiones extraídas del estudio de los distintos tipos de NAT y de los requisitos propuestos por el IETF para la mejora de las comunicaciones en este escenario.

2.2 ¿Qué es NAT?

La siguiente definición la encontramos en IP Network Address Translator (NAT) Terminology and Considerations [RFC 2663], apartado 3:

¿Que es NAT?

La traducción de dirección de red (Network Address Translation), es un método por el cual las direcciones IP son mapeadas desde un espacio de direcciones a otro, proporcionando enrutamiento transparente en los extremos finales. Hay muchas variantes para la traducción de direcciones que se prestan a diferentes usos. Sin embargo, todos los dispositivos NAT deberían compartir las características siguientes:

a) Traducción de direcciones transparente.
b) Enrutamiento transparente por medio de traducción de direcciones. (Enrutamiento como reenvío de paquetes y no intercambio de información de enrutamiento).
c) Traducción de paquetes de error ICMP.

A continuación se muestra un diagrama ilustrando un escenario en el cual un NAT esta habilitado en un enrutador de borde de dominio, conectado a Internet a través del enrutador del proveedor de servicio.

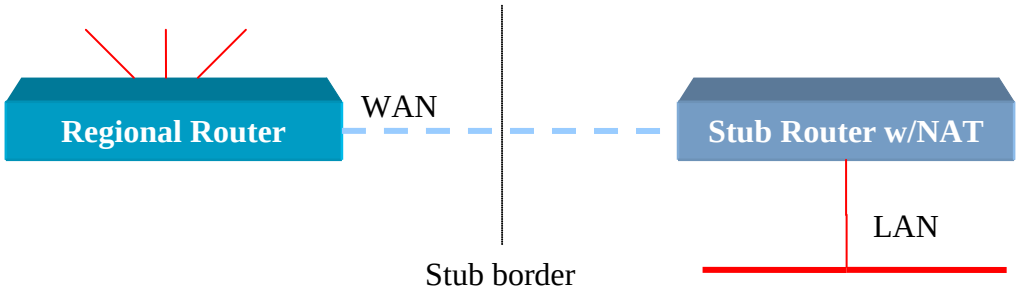

Figura 1. Típico escenario de operativa de NAT

NAT tiene muchas formas de funcionamiento, de hecho la clasificación de los diferentes tipos de NAT es actualmente una tarea muy compleja debido principalmente a la ausencia de estándares y la falta de acuerdos entre los fabricantes. Siendo poco rigurosos, se podría presentar una primera clasificación como la propuesta en Wikipedia, en la que se distinguen los siguientes tipos:

Estático

Es un tipo de NAT en el que una dirección IP privada se traduce a una dirección IP pública, y donde esa dirección pública es siempre la misma. Esto le permite a un host, como un servidor Web, el tener una dirección IP de red privada pero aún así ser visible en Internet.

Dinámico

Es un tipo de NAT en la que una dirección IP privada se mapea a una IP pública basándose en una tabla de direcciones IP registradas (públicas). Normalmente, el router NAT en una red mantendrá una tabla de direcciones IP registradas, y cuando una IP privada requiera acceso a Internet, el router elegirá una dirección IP de la tabla que no esté siendo usada por otra IP privada. Esto permite aumentar la seguridad de una red dado que enmascara la configuración interna de una red privada y lo que dificulta a los hosts externos de la red el poder ingresar a ésta. Para este método se requiere que todos los hosts de la red privada que deseen conectarse a la red pública posean al menos una IP pública asociada.

Sobrecargado

La forma más utilizada de NAT, proviene del NAT dinámico, ya que toma múltiples direcciones IP privadas (normalmente entregadas mediante DHCP) y las traduce a una única dirección IP pública utilizando diferentes puertos. Esto se conoce también como **PAT** (**Port Address Translation - Traducción de Direcciones por Puerto**), NAT de única dirección o NAT multiplexado a nivel de puerto.

Solapado

Cuando las direcciones IP utilizadas en la red privada son direcciones IP públicas en uso en otra red. El router posee una tabla de traducciones en donde se especifica el reemplazo de éstas con una única dirección IP pública. Así se evita los conflictos de direcciones entre las distintas redes.

Pero esta clasificación inicial de los diferentes tipos de NAT es muy poco rigurosa. En los apartados siguientes veremos como el grupo behave ha profundizado en esta sentido, observando minuciosamente como se comportan las cajas NAT en situaciones concretas de intercambio de información.

2.3 Asignación de IP y puerto

Cuando un extremo privado abre una sesión a través del NAT, este asigna a la sesión una dirección IP y un puerto, de manera que las sucesivas respuestas desde el extremo público puedan ser recibidas por el NAT, transformadas y reenviadas al extremo privado. Se produce en este caso una transformación:

f: (IP privada, Puerto privado) ←→ (IP pública, Puerto público)

Para la mayoría de las aplicaciones es imposible distinguir el funcionamiento del NAT cuando hay múltiples sesiones simultáneas con diferentes extremos en la red pública.

La clave se encuentra en descifrar como se reutilizan los mapeos establecidos para nuevas sesiones, después de haber generado un mapeo de direcciones entre una tupla interna (X:x) y una externa (Y:y). Asumimos que al extremo (X: x) privado, se le asocia (X1',x1') para la primera sesión. Ahora el extremo (X: x) envía un mensaje hacia (Y2, y2) y obtiene la asociación con (X2', x2') en el NAT. Las relaciones entre (X1', x1') y (X2', x2') para las diferentes combinaciones de (Y1,y1) y (Y2,y2), son determinantes para describir el funcionamiento del NAT. Gráficamente el escenario es el que se muestra en la siguiente figura:

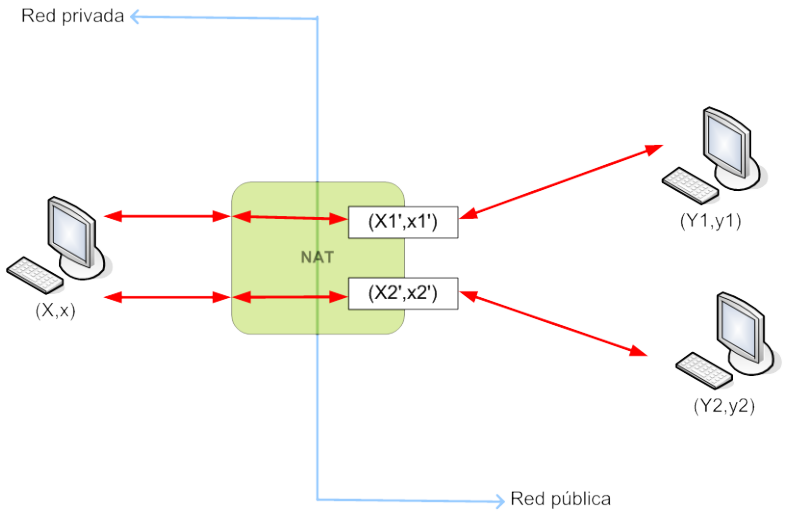

Figura 2. Escenario para la comprensión de la asignación de IP y puerto

La clasificación en función de la asignación de direcciones del NAT se muestra en los siguientes apartados.

2.3.1 Independiente del extremo

El NAT reutiliza la primera asociación para todos los paquetes enviados desde la misma tupla interna (X, x) hacia cualquier tupla externa.

(X1', x1') = (X2', x2') para cualquier (Y2, y2)

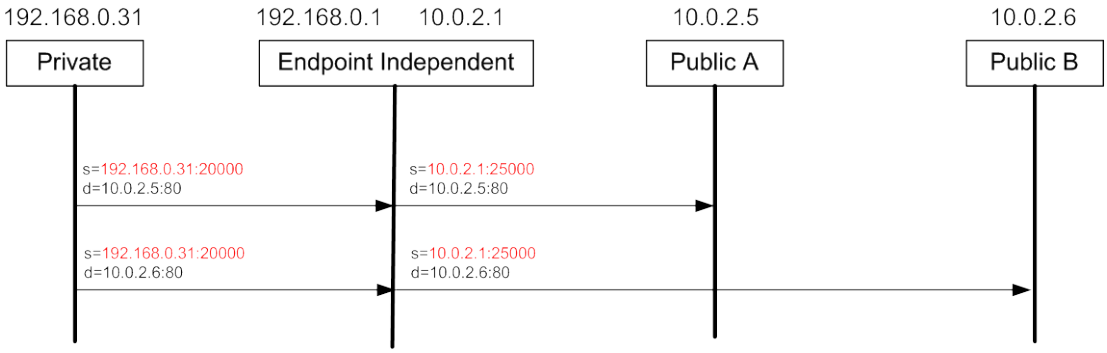

Figura 3. Secuencia NAT independiente del extremo

En el ejemplo se puede observar como un cliente en la red privada desea comunicarse con dos máquinas públicas utilizando la misma tupla IP-Puerto en el origen.

El NAT reutiliza la primera asociación en las restantes conexiones de forma que la tabla de asignación quedaría de la siguiente manera:

Red Privada	Red Pública	Asociación
192.168.0.31:20000	10.0.2.5:80	10.0.2.1:25000
192.168.0.31:20000	10.0.2.6:80	10.0.2.1:25000

Tabla 1. Tabla asignación NAT independiente del extremo

2.3.2 Dependiente de la dirección

El NAT reutiliza la primera asociación para posteriores envíos desde la misma tupla interna (X, x) hacia la misma dirección IP externa sin tener en cuenta el número de puerto.

(X1', x1') = (X2', x2') si y solo si Y1 = Y2

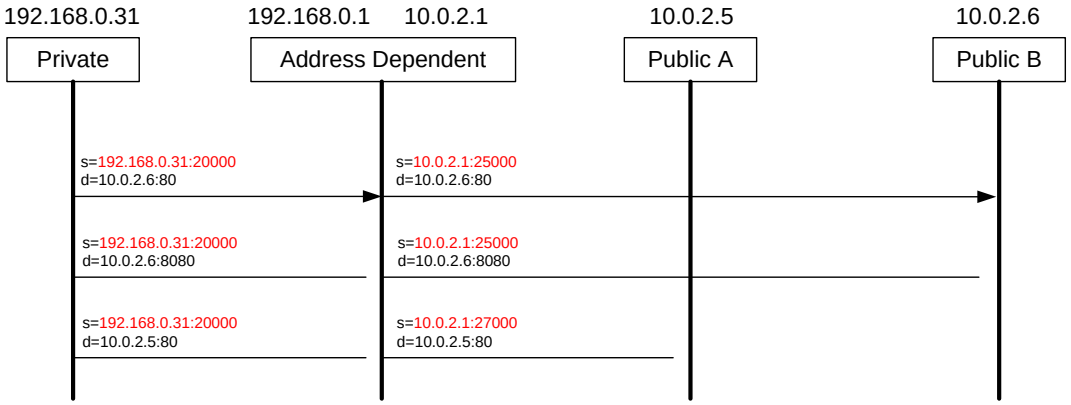

Figura 4. Secuencia NAT dependiente de la dirección

La tabla de asignaciones del NAT es la que se muestra a continuación:

Red Privada	Red Pública	Asociación
192.168.0.31:20000	10.0.2.6:80	10.0.2.1:25000
192.168.0.31:20000	10.0.2.6:8080	10.0.2.1:25000
192.168.0.31:20000	10.0.2.5:80	10.0.2.1:27000

Tabla 2. Tabla asignación NAT dependiente de la dirección

2.3.3 Dependiente de la dirección y del puerto

El NAT reutiliza la primera asociación para posteriores envíos desde la misma tupla interna (X, x) hacia la misma tupla externa, mientras esta asociación siga activa.

(X1', x1') = (X2', x2') si y solo si (Y1, y1) = (Y2, y2)

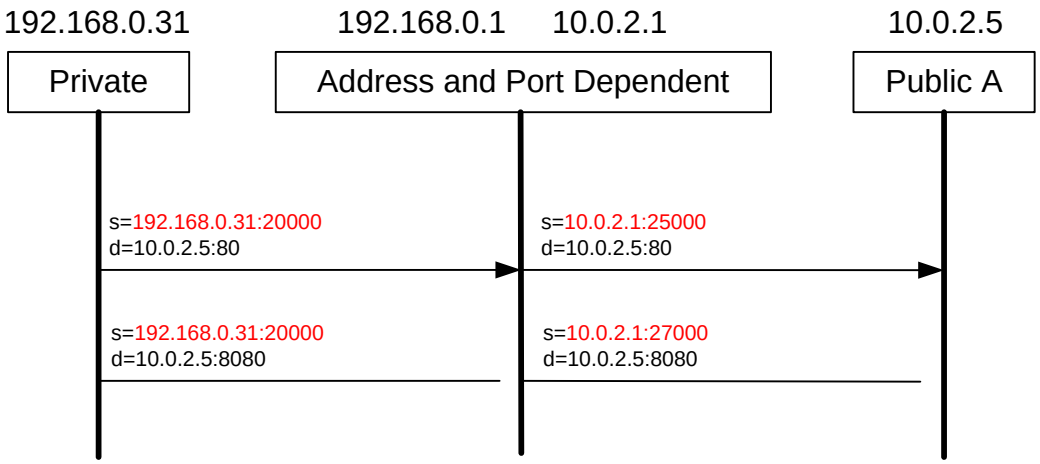

Figura 5. Secuencia NAT dependiente de la dirección y el puerto

La tabla de asignación correspondiente al escenario mostrado contendría las siguientes entradas:

Red Privada	Red Pública	Asociación
192.168.0.31:20000	10.0.2.5:80	10.0.2.1:25000
192.168.0.31:20000	10.0.2.5:8080	10.0.2.1:27000

Tabla 3. Tabla asignación NAT dependiente de la dirección y el puerto

2.4 Asignación con parque de direcciones IP externas

Para aquellos NAT con un parque de direcciones IP externas, se puede hacer la siguiente clasificación.

2.4.1 Arbitraria

Asignan las direcciones IP de forma arbitraria (ej. al azar). De esta forma, una dirección IP interna puede tener asociadas varias direcciones IP externas simultáneamente para diferentes sesiones.

Para estos NAT, aquellas aplicaciones que usan varios puertos desde el mismo extremo pueden tener problemas si no negocian las direcciones IP individualmente (ej. algunas aplicaciones que usan **RTP** y **RTCP**).

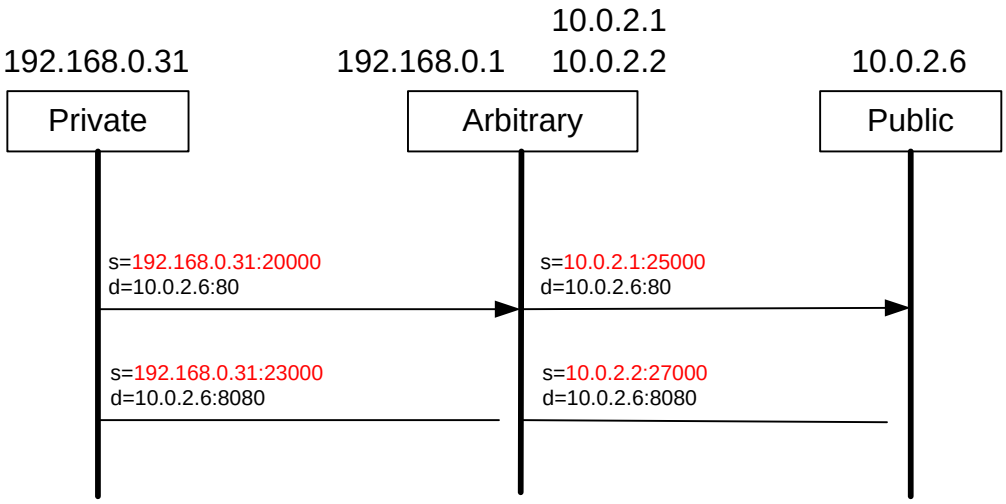

Figura 6. Secuencia NAT asignación arbitraria de IP

Red Privada	Red Pública	Asociación
192.168.0.31:20000	10.0.2.6:80	10.0.2.1:25000
192.168.0.31:23000	10.0.2.6:8080	10.0.2.2:27000

Tabla 4. Tabla asignación IP arbitraria

2.4.2 Pareada

Son aquellos NAT que asignan la misma dirección IP externa para todas las sesiones abiertas desde la misma dirección IP interna.

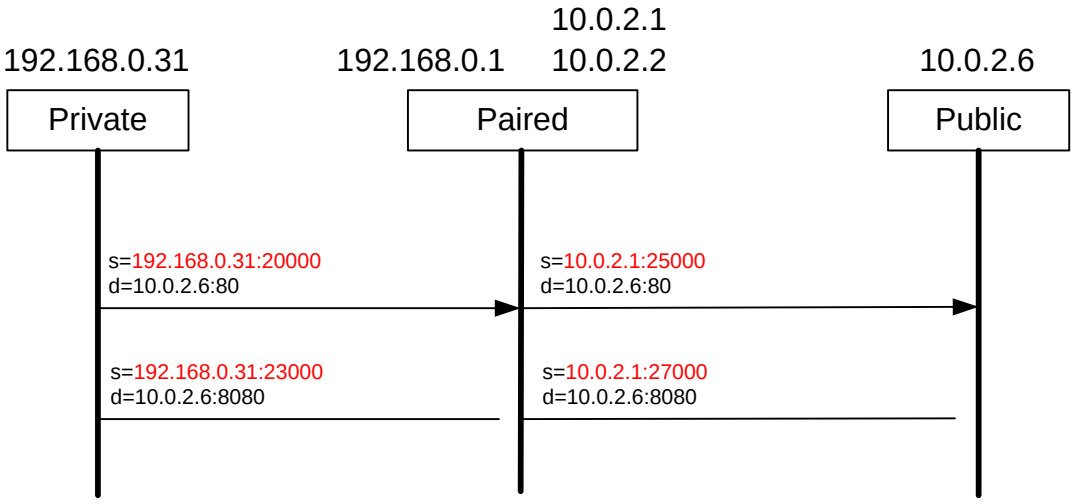

Figura 7. Secuencia NAT asignación pareada de IP

Red Privada	Red Pública	Asociación
192.168.0.31:20000	10.0.2.6:80	10.0.2.1:25000
192.168.0.31:23000	10.0.2.6:8080	10.0.2.1:27000

Tabla 5. Tabla asignación IP pareada

2.5 Asignación del puerto

Como es sabido, la numeración de los puertos se divide en los siguientes rangos:

Bien conocido (0-1023)
Registrado (1024-49151)
Dinámico/Privado (49152-65535)

Para la mayoría de las aplicaciones estos puertos son puertos de destino, por lo que el mapeo de un puerto origen a otro puerto ya registrado probablemente no causará ningún problema.

Algunos NAT utilizan exclusivamente los puertos dinámicos. Otros NAT utilizan todos excepto los bien conocidos y posiblemente asignen primero los dinámicos. Hay NAT que conservan el número de puerto si este es bien conocido.

En User Datagram Protocol [RFC 0768] se especifica que el puerto origen sea el 0 si no se esperan mensajes de respuesta. En este caso da lo mismo que puerto asigne el NAT puesto que este puerto no se usará.

Para explicar como se comportan los diferentes tipos de NAT a la hora de asignar el puerto externo, utilizaremos el siguiente esquema:

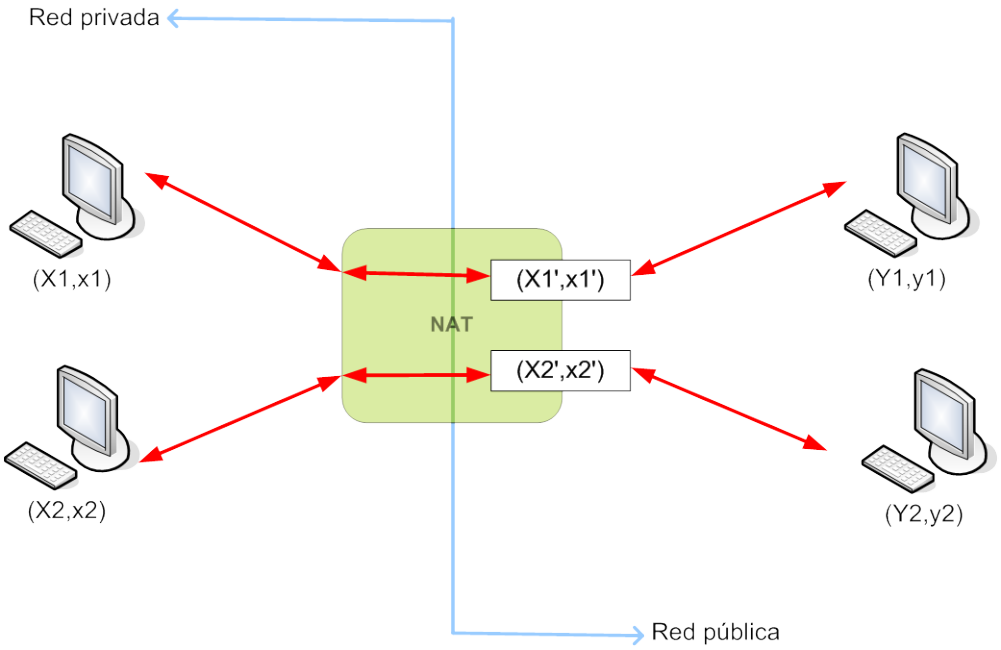

Figura 8. Escenario para la comprensión de la asignación del puerto

2.5.1 Conservativa

Algunos NAT intentan preservar el número de puerto usado internamente en la red privada cuando asocian la tupla (IP-Puerto) interna con la tupla externa.

x1 = x1' y x2=x2'

En caso de colisión (x1 = x2), las posibilidades de funcionamiento son múltiples. Algunos NAT sobrescriben el mapeo anterior para conservar el mismo puerto. Otros NAT asignan una dirección IP diferente del parque de direcciones externas. De todas formas, si el puerto esta siendo utilizado en todas las direcciones IP externas disponibles, se asignará un puerto diferente (y por lo tanto no se puede asegurar la conservación del puerto).

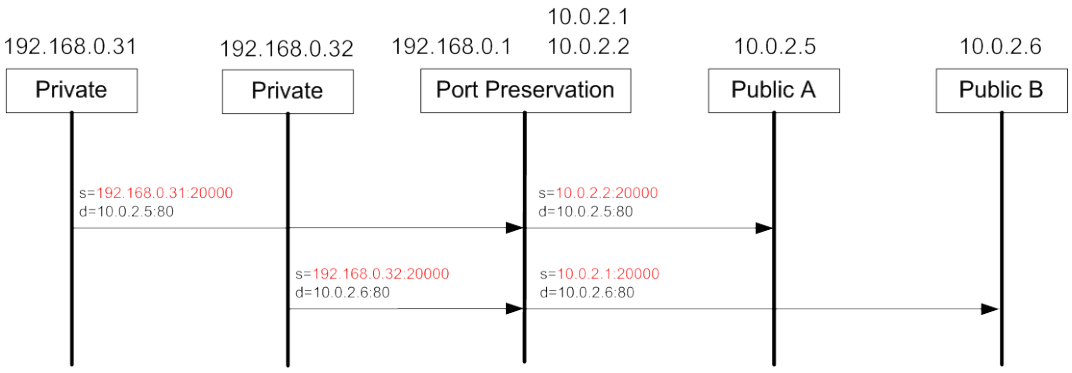

Figura 9. Secuencia NAT asignación conservativa de puerto

2.5.2 No conservativa

Este tipo de NAT no conserva el puerto origen de la comunicación en la asignación del puerto externo.

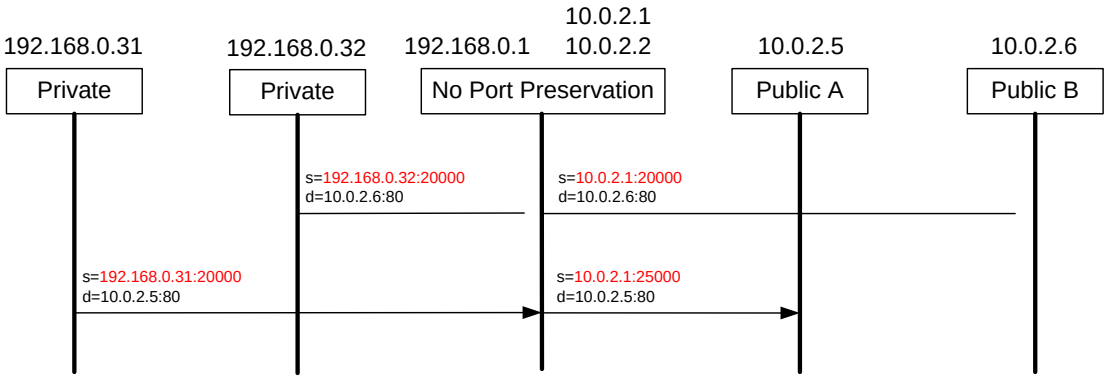

Figura 10. Secuencia NAT asignación no conservativa de puerto

2.5.3 Sobrecargada

El puerto se mantiene siempre, incluso cuando no hay direcciones IP disponibles en el NAT.

X1' = X2' y x1=x2=x1'=x2'

Es evidente que muchas aplicaciones fallarán si el NAT utiliza sobrecarga de puertos.

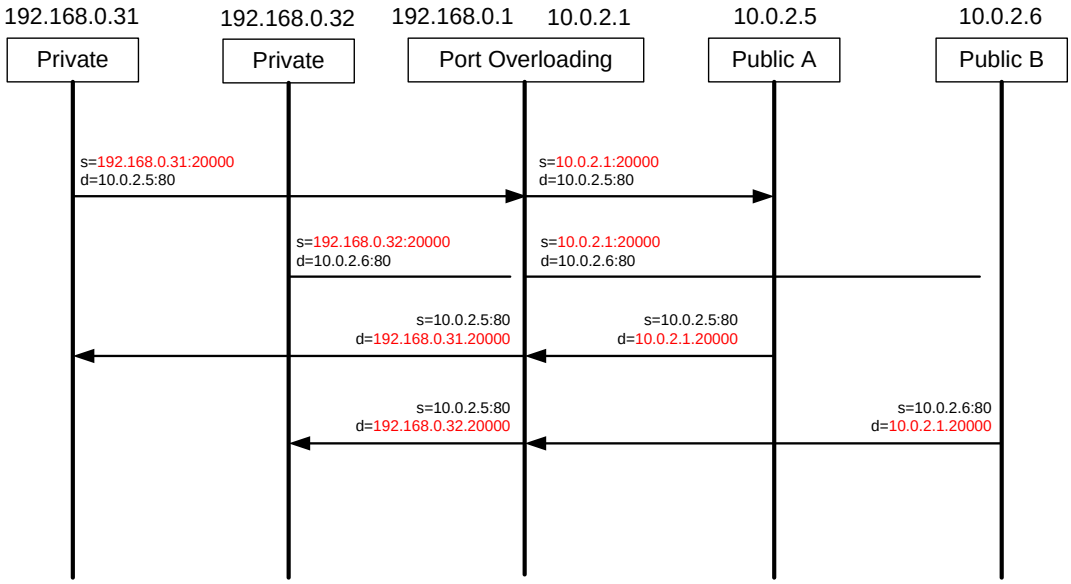

Figura 11. Secuencia NAT asignación de puerto sobrecargada

En este caso el NAT retransmite sobre el origen de la respuesta.

Red Privada	Red Pública	Asociación
192.168.0.31:20000	10.0.2.5:80	10.0.2.1:20000
192.168.0.32:20000	10.0.2.6:80	10.0.2.1:20000

Tabla 6. Secuencia NAT asignación de puerto sobrecargada

2.5.4 Conservando la paridad

Algunos NAT conservan la paridad de la numeración de los puertos. Así un puerto par se mapeará a un puerto par y uno impar a otro impar. La [RFC 3550] señala que RTP utiliza puertos pares y RTCP puertos impares. Además está permitido que estos puertos sean cualesquiera y definidos de manera independiente (pueden ser no contiguos [RFC 3605]).

Algunas implementaciones no incluyen las directivas de la [RFC 3605] y no reconocen cuando el otro extremo ha especificado el puerto RTCP independientemente del puerto RTP. Si estas implementaciones reciben un puerto RTP impar, quizás después de haber sido transformado por un NAT, y a continuación siguiendo la [RFC 3550], cambian el puerto RTP al siguiente puerto par, obviamente se perderá la comunicación RTP. Por lo tanto, conservando la paridad de puertos se posibilita que las aplicaciones peer-to-peer que no soporten la especificación explícita de los puertos RTP y RTCP establezcan la comunicación.

2.5.5 Contigua

Algunos NAT intentan conservar la contigüidad de los puertos según la regla RTCP=RTP+1. Asumen que la aplicación abrirá un puerto RTP y a continuación otro para RTCP. En realidad los NAT no pueden distinguir entre un paquete RTP y cualquier otro tipo de paquete UDP.

Por lo tanto es necesario que las aplicaciones negocien los puertos RTP y RTCP por separado.

2.6 Refresco de mapeo

Existen grandes diferencias entre los tiempos que utilizan los NAT para mantener un mapeo activo mientras no haya tránsito de paquetes.

Algunos NAT mantienen el mapeo activo cuando un paquete viaja de la zona interna hacia la zona externa del NAT. A este comportamiento se le llama “refresco de salida”.

Otros NAT mantienen el mapeo activo cuando un paquete es enviado desde el exterior hacia el interior del NAT. Este comportamiento es llamado “refresco de entrada”.

También hay NAT que mantienen el mapeo activo en ambos casos.

El refresco de salida es necesario para que el cliente pueda mantener el mapeo activo.

El refresco de entrada puede ser necesario en aplicaciones que no tienen tráfico hacia el exterior. Claramente esto puede suponer un riesgo de seguridad en tanto que se pueden recibir ataques externos.

2.7 Conflictos en el espacio de direcciones

La mayor parte de los NAT y particularmente los de usuario final obtienen sus direcciones IP, la dirección del enrutador por defecto y otra información sobre la configuración IP desde una red externa. Por otro lado, el enrutador inicializa su red interna con una de las direcciones del espacio de direcciones IP privadas asignadas a este propósito en la [RFC1918].

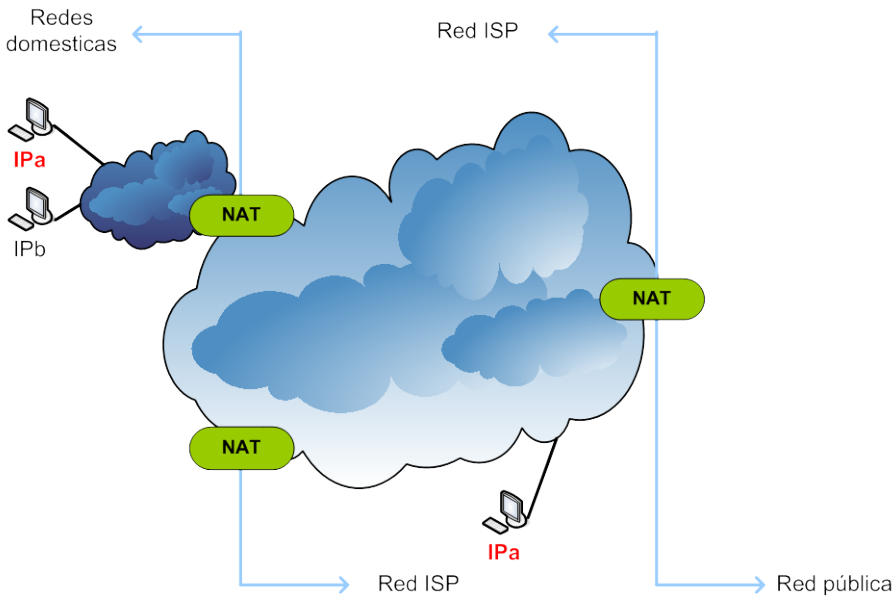

Figura 12. Escenario de NAT en cascada con conflicto de direcciones IP

En este escenario no hay ninguna garantía de que haya conflictos de direcciones IP entre la red interna y la red externa.

Los NAT deben ser robustos frente a este conflicto. Hay dos posibles soluciones:

La primera alternativa pasa por la elección de una red interna que no entre en conflicto con la red externa, si esta está configurada con una dirección IP privada indicada por la [RFC 1918]. La desventaja de este método es que si la red externa cambia

dinámicamente, el NAT podría tener que reconfigurar su red interna para no colisionar con la nueva configuración.

La segunda solución es que el NAT pueda enviar y recibir tráfico correctamente aun cuando los interfaces interno y externo se solapen. En general, el NAT debe permitir la comunicación entre todos los nodos internos con todos los externos que tengan direcciones IP públicas (no definidas en [RFC 1918]) o direcciones privadas que no entren en conflicto con las direcciones usadas en su red privada.

Si las subredes interna y externa están configuradas con subredes solapadas, no hay forma posible de comunicar un nodo interno con IP definida en la [RFC 1918], con un nodo externo que tenga la misma dirección IP. Como veremos más adelante, estos nodos pueden usar técnicas de traspaso de NAT con la ayuda de un servidor STUN con una dirección IP pública (no definida en la [RFC 1918]).

2.8 Filtrado de paquetes de entrada

Cuando un nodo interno abre una sesión de salida a través del NAT, este asigna una regla de filtrado para el mapeo entre la tupla IP-puerto interna (X: x) y la externa (Y: y). Dependiendo de cómo el NAT filtre los paquetes procedentes desde el exterior, se pueden clasificar los NAT como se describe en este apartado.

2.8.1 Independiente del extremo

El NAT filtra los paquetes cuyo destino no sea (X: x), independientemente de la dirección y puerto de origen (Z: z). El NAT reenvía cualquier paquete destinado a (X: x). En otras palabras, enviar un paquete desde el lado interno del NAT hacia el lado externo, es suficiente para permitir la entrada de cualquier paquete desde el exterior hacia el interior (X: x).

2.8.2 Dependiente de la dirección

El NAT filtra los paquetes que no están destinados hacia la tupla interna (X: x). Adicionalmente, el NAT filtra los paquetes destinados A (X: x) procedentes de (Y: y) si el extremo (X: x) no ha enviado ningún paquete a Y independientemente del puerto usado en el envío. En otras palabras, para recibir paquetes de un extremo externo, es necesario que el extremo interno envíe antes algún paquete a la dirección IP externa (a cualquier puerto).

2.8.3 Dependiente del puerto y la dirección

Es similar al anterior excepto porque el puerto externo al que se envían los paquetes también es relevante. El NAT filtra paquetes que no están destinados a la dirección IP y

puerto internos (X: x). Además, filtrará los paquetes con destino (X: x) y origen (Y: y) si previamente no se han realizado envíos desde (X: x) hacia (Y: y). Es decir, para poder recibir paquetes desde un extremo externo, se ha de realizar previamente algún envío de paquetes desde el extremo interno (misma IP y puerto).

En teoría, el filtro basado en IP y puerto es más seguro que el basado únicamente en la dirección IP, puesto que el extremo externo pueden en realidad ser nodos diferentes que se encuentran detrás de un NAT y uno de ellos puede resultar ser un ataque. Desgraciadamente esta política interfiere con aplicaciones que esperan recibir paquetes UDP en más de un puerto UDP.

2.9 Funcionamiento en horquilla

Si dos máquinas, X1 y X2, se encuentran detrás del mismo NAT e intercambian tráfico de paquetes UDP, el NAT puede reservar una dirección en la cara externa del NAT para X2, (X2', x2').

Si X1 envía paquetes UDP hacia (X2', x2'), el NAT deberá reenviar los paquetes hacia X2. Este funcionamiento se ilustra en la siguiente figura:

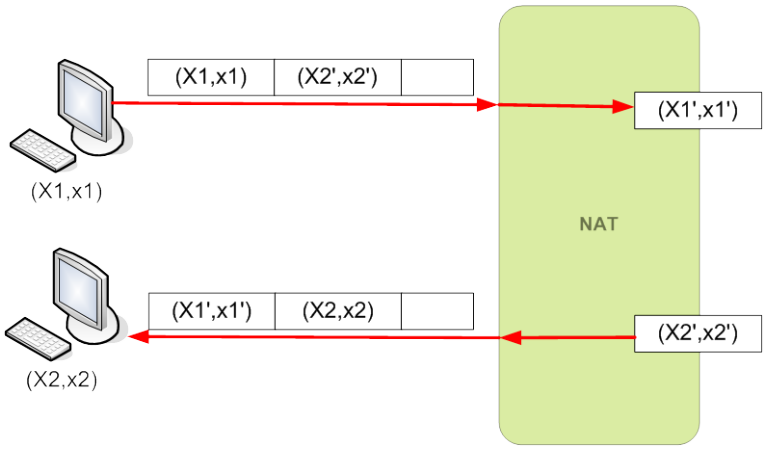

Figura 13. Escenario de funcionamiento en horquilla

En este caso el NAT puede presentar los paquetes con la IP y puerto interna (X1, x1) o la IP y puerto externa (X1', x1'). Si utiliza la dirección IP y el puerto internos, se pueden ocasionar problemas en aplicaciones que esperan una dirección IP y un puerto externos.

2.10 Puertas a nivel de aplicación (ALG Aplication Layer Gateway)

Algunos NAT implementan ciertos ALGs para determinadas aplicaciones. Estos ALGs pueden estar activados permanentemente, otros NAT los tienes activados por defecto pero permiten desactivarlos y otros los tienen desactivados por defecto pero permiten activarlos.
Estos ALGs pueden interferir con los métodos UNSAF o con los protocolos que intentan ser conscientes de los NAT, y por lo tanto deben emplearse con mucho cuidado.

2.11 Propiedades determinísticas

La clasificación de los NAT es muy complicada por el hecho de que bajo algunas circunstancias algunos NAT cambian su forma de actuar. Se pueden observar NAT que conservan el puerto o que implementan algoritmos para seleccionar un puerto libre en lugar de otro que esté libre en ese momento. Si el puerto externo elegido esta siendo utilizado por otra sesión, el NAT debe seleccionar un puerto diferente. Este escenario provoca que el NAT se comporte de maneras inesperadas dependiendo de la situación inicial.

Si por ejemplo tenemos tres máquinas intentando enviar datos por el mismo puerto a través de un NAT que intenta conservar el puerto y se dispone de una sola dirección IP (llamémosla X1'), el primero que envía (ej. X1) obtendrá el puerto externo x, pero los siguientes obtendrán los puertos x2' y x3', diferentes de x. Hay NAT que cambian las características de mapeo y filtrado entre (X1: x) y (X2: x). Para empeorar las cosas, podemos encontrar NAT que mantienen estas características entre (X1: x) y (X2: x), pero cambian para la tercera comunicación por el puerto x.

En general un NAT no determinístico cambia la forma de actuar cuando se produce algún conflicto, Ej. Cuando el puerto que utiliza habitualmente esta siendo utilizado por otro mapeo.

El funcionamiento normal en cuanto a la asignación de dirección IP y puerto y al filtrado de paquetes se denomina comportamiento primario. El funcionamiento que aparece después del primer conflicto se denomina comportamiento secundario y el funcionamiento después del segundo conflicto se denomina comportamiento terciario.

2.12 Destino inalcanzable ICMP

Cuando un NAT envía un paquete hacia la red externa, es posible que reciba un mensaje ICMP como respuesta a este. Esta respuesta puede haberla enviado el extremo al que iba destinado o cualquier enrutador por el que transite.

El NAT no debería filtrar los mensajes ICMP basándose en su dirección IP de origen. En su lugar, el NAT debería reescribir estos mensajes ICMP y reenviarlos al nodo interno o externo apropiado. Este reenvío de paquetes ICMP se realizará mientras el mapeo UDP

esté activo. Así mismo, la recepción de un paquete ICMP no debe destruir el mapeo existente.

Bloquear paquetes ICMP de destino inalcanzable no aporta ventajas significativas en cuanto a la seguridad. En realidad, al interceptar este tipo de paquetes interfiere en aplicaciones como traceroute o de descubrimiento de MTU sobre UDP.

2.13 Fragmentación de paquetes de salida

Cuando la MTU del enlace externo es demasiado pequeña, puede suceder que sea necesario fragmentar los paquetes enviados desde el lado interno hacia el lado externo del NAT.

Los paquetes podrían tener el bit DF (Don´t Fragment) a 1 o a 0.

2.14 Recepción de paquetes fragmentados

Un NAT puede recibir paquetes fragmentados. Dependiendo de las condiciones de la red, el paquete que contiene la cabecera puede llegar en cualquier fragmento.

Un NAT puede ser capaz de recibir solo los fragmentos en orden (el paquete con la cabecera es el primero en llegar) y reenviar esos paquetes hacia el nodo interno.

Existen NAT que son capaces de recibir los fragmentos en orden y fuera de orden y reenviar los fragmentos individualmente o en un paquete re ensamblado hacia el nodo interno.

Otros NAT no poseen ninguna de estas capacidades.

2.15 Requisitos NAT

A continuación se enumeran los requisitos que aparecen en el documenta Network Address Translation (NAT) Behavioral Requirements for Unicast UDP [RFC 4787], elaborado por el grupo de trabajo Behave del IETF.

Requisito 1

Los NAT **deben** mapear las asociaciones como **independiente del extremo**.

Requisito 2

Es **recomendable** que los NAT tengan un modo **pareado** si disponen de un parque de direcciones IP externas.

Requisito 3

El NAT **no debe** usar la **sobrecarga de puertos**.

- Si el puerto origen está en el rango (0-1023) es **recomendable** que el puerto origen del NAT esté en el mismo rango.
- Si el puerto origen esta en el rango (1024-65535) es **recomendable** que el puerto origen del NAT esté en el mismo rango.

Requisito 4

Es **recomendable** que el NAT **conserve la paridad** en la numeración de los puertos.

Requisito 5

Un mapeo UDP **no debe** expirar en menos de dos minutos, a menos que se deba aplicar la excepción a este requisito.

- Para puertos destino bien conocidos **puede** expirar en un tiempo menor que el especificado por la aplicación que corre sobre ese puerto de destino.
- El valor de este tiempo en el NAT **puede** ser configurable.
- El valor **recomendable** por defecto para un mapeo UDP en el NAT es de cinco minutos.

Requisito 6

El refresco de un mapeo en el NAT **debe ser** de **salida**.

- Adicionalmente el refresco **puede ser** de entrada.

Requisito 7

Aquellos NAT con interfaces externos configurables dinámicamente, **deben:**

- Asegurar que su red interna utiliza direcciones IP que no colisionan con la red externa.
- Ser capaces de enviar y recibir tráfico entre todos los nodos internos y todos los nodos externos cuyas direcciones IP entran en conflicto con la red interna.

Requisito 8

Si se considera de mayor importancia la **transparencia** de las aplicaciones, es **recomendable** que el NAT realiza un filtro **independiente del extremo**. Si es más importante que el NAT sea más **riguroso** a la hora de filtrar los paquetes, es **recomendable** que el NAT realice un filtro **dependiente de la dirección**.

- La configuración del nivel de filtro puede ser configurable por el administrador del NAT.

Requisito 9

El NAT **debe** soportar el funcionamiento en **horquilla**.

- El NAT **debe** presentar los paquetes en horquilla con la dirección IP y el puerto **externos.**

Requisito 10

Para eliminar las interferencias con los métodos UNSAF y permitir la integración total de comunicaciones UDP, los ALGs para protocolos basados en UDP **deben** estar desactivados.

- Si el NAT incluye ALGs, es **recomendable** que el administrador del NAT pueda habilitar o deshabilitar cada ALG por separado.

Requisito 11

Un NAT **debe** tener un comportamiento **determinístico**, es decir, **no debe** cambiar la asignación de IP y puerto ni el filtrado de paquetes en ningún momento ni bajo ninguna circunstancia.

Requisito 12

La recepción de cualquier tipo de paquete ICMP **no debe** finalizar el mapeo del NAT.

- La configuración por defecto del NAT **no debería** filtrar los paquetes ICMP basándose en la dirección IP origen.
- Es **recomendable** que el NAT soporte mensajes ICMP de destino inalcanzable.

Requisito 13

Si se recibe un paquete de una dirección IP interna con el bit DF a 1 y es necesario fragmentar, el NAT **debe** responder con un mensaje ICMP indicando que “es necesaria la fragmentación y por lo tanto la activación del bit DF”.

- Si el paquete ya tiene el bit DF a 0, el NAT **debe** fragmentar los paquetes y **debería** enviar los fragmentos en orden.

Requisito 14

El NAT **debe** ser capaz de recibir paquetes UDP fragmentados en orden y fuera de orden.

- Un NAT que es capaz de recibir los fragmentos fuera de orden, **debe** estar diseñado de forma que los ataques DoS basados en la fragmentación no comprometan la capacidad de procesar los paquetes IP fragmentados en orden y los no fragmentados.

2.16 Conclusión

Del estudio de los distintos modos de funcionamiento de NAT nos damos cuenta de la necesidad de estandarizar los mecanismos de asignación de direcciones con dos objetivos fundamentales.

El primer objetivo se debe centrar en la homogeneidad del comportamiento de las cajas NAT para facilitar y simplificar el desarrollo de aplicaciones, maximizando en la medida de lo posible el número usuarios que pueden operar a través del NAT.

El segundo objetivo, aunque no menos importante, se deberá dirigir a la modernización del mecanismo NAT para adecuarlo a los nuevos usos de Internet (como aplicaciones de voz sobre IP o protocolos de intercambio de información)

Por otra parte es importante darse cuenta de que el mecanismo NAT con su simplicidad es capaz de ofrecer una alternativa, aunque solo sea temporal, a la implantación de IPv6 y que salvo en algunas aplicaciones ofrece una solución satisfactoria a la escasez de direcciones IPv4. Por lo tanto, aún cuando este mecanismo tiene deficiencias y plantea problemas en determinados usos, es una alternativa sencilla y muy útil que se ha utilizado de manera extensiva y ha demostrado su validez en muchísimos casos.

Finalmente se muestra una tabla de comportamiento del NAT ideal en función de los requisitos extraídos por el grupo de trabajo behave del área de trasporte del IETF:

CARACTERISTICA	**FUNCIONAMIENTO**	**NECESIDAD**
MAPEO	Independiente del extremo	Obligatorio
PARQUE IPs EXTERNAS	Asignación pareada	Recomendable
SOBRECARGA DE PUERTOS	No	Obligatorio
	Conservar rangos	Recomendable
PARIDAD DE PUERTOS	Si	Recomendable
EXPIRACION MAPEO UDP	Mínimo 2 minutos	Obligatorio
	Puertos bien conocidos, menor de 2 minutos	Posible
	Tiempo configurable	Posible
	Al menos 5 minutos	Recomendable
	Refresco de salida	Obligatorio
	Refresco de entrada	Posible
FILTRO	Independiente del extremo	Recomendable
	Dependiente de dirección	Recomendable
	Configurable	Posible
HORQUILLA	Presentar paquetes en horquilla con dirección externa	Obligatorio
ALGs	Desactivados	Obligatorio

	Administrados	Recomendable
COMPORTAMIENTO (MAPEO Y FILTRO)	Deterministico	Obligatorio
RECEPCIÓN ICMP	No termina mapeo	Obligatorio
	No filtra basandose en IP origen	Obligatorio
	Soporte de mensaje de destino inalcanzable	Obligatorio
FRAGMENTACION	Soporte de envío en orden	Obligatorio
	Recepción en orden y fuera de orden	Obligatorio

Tabla 7. Tabla comportamiento de NAT ideal

Capítulo 3

Análisis

3.1 Introducción

En este capítulo se detalla el proceso de análisis del problema y se presenta una solución a nivel lógico de las entidades que deberá implementar el sistema para la consecución de los objetivos marcados en el proyecto.

Recordando los objetivos principales, se quiere construir una herramienta que permita el estudio del comportamiento de NAT y sus problemas derivados. Para ello se considera imprescindible la creación de los tres elementos fundamentales que forman parte de un escenario básico de conexión a través de NAT sobre protocolo UDP. Estos son, un cliente que se encuentra en la zona privada de una red conectada a Internet a través de un NAT, el NAT propiamente dicho y finalmente un servidor que se encuentra en una red pública (la zona pública del NAT).

Para la detección de NAT y su comportamiento, se ha seleccionado el protocolo STUN y por lo tanto será necesario implementar los rudimentos básicos de un cliente y un servidor STUN. Esto quiere decir que no se pretende la implementación completa de dicho protocolo, sino únicamente aquellos aspectos encaminados a la detección de NAT y no las capacidades de control de sesión, control de conexión, detección de errores, etc.

En cualquier caso, aquellas funciones que se implementen tanto para el cliente como para el servidor STUN seguirán las reglas de implementación descritas en la [RFC 3489]. Siguiendo estas reglas de implementación, los test generados para la plataforma de pruebas serán útiles en una plataforma real.

En primer lugar se describen brevemente algunas características del protocolo STUN.

A continuación se muestra el diagrama lógico de un NAT configurable para que abarque el mayor número de comportamientos posible, emulando así distintos tipos de NAT en cuanto a su funcionamiento.

Por último se expone un diagrama lógico completo de la solución, en el que intervienen todas las partes involucradas y se muestra como deben colaborar para conseguir el objetivo de simulación completa.

3.2 STUN

Session Traversal Utilities for NAT (STUN), es un protocolo de red del tipo cliente/servidor que proporciona funciones que permiten a entidades situadas detrás de un NAT conocer la dirección asignada por el NAT y ser capaces de mantener esta dirección abierta el tiempo que la necesiten. STUN está definido en [RFC 3489].

Un **cliente** STUN es una entidad que genera **peticiones** STUN y recibe **respuestas** STUN. También pueden generar **indicaciones** STUN

Un **servidor** STUN es una entidad que recibe **peticiones** STUN y genera **respuestas** STUN. También puede generar **indicaciones** STUN.

Una **dirección de transporte** es una combinación de dirección IP y puerto de protocolo de trasporte (como TCP o UDP)

Una **dirección de transporte reflexiva** es aquella dirección que identifica a un cliente visto desde otro nodo en una red IP. Cuando existe un NAT entre el cliente y el host, representa el vínculo asociado al cliente en la zona pública del NAT. El cliente puede conocer esta dirección mediante el atributo MAPPED-ADDRESS o XOR-MAPPED-ADDRESS de una repuesta STUN.

STUN soporta dos tipos de transacciones.

El primer tipo son transacciones petición/respuesta, en las que el cliente envía una petición a un servidor y el servidor retorna una respuesta.

El segundo tipo son indicaciones iniciadas por el servidor o el cliente y no provocan una respuesta.

Las peticiones de vínculo (Binding Requests), son enviadas desde el cliente hacia el servidor. Cuando la petición llega al servidor STUN, puede que haya pasado a través de uno o varios NATs, situados entre el cliente y el servidor. En este caso, la dirección de trasporte del origen de la petición recibida por el servidor, coincidirá con la dirección pública del NAT más cercano al servidor STUN. El servidor STUN copia esta dirección dentro de la respuesta y la envía. Cada tipo diferente de NAT enrutará esta respuesta de manera que alcance al cliente STUN. De esta forma, el cliente STUN conoce su dirección de trasporte utilizada por el NAT más lejano que se encuentra entre este y el servidor STUN.

3.2.1 Estructura del mensaje

Los mensajes STUN son del tipo TLV (Tipo-Longitud-Valor), codificados en binario con formato “big endian” (es decir, primero el byte más significativo). Las constantes numéricas van en decimal (base 10). Todos los mensajes constan de una cabecera y un cuerpo. El cuerpo esta formado por cero o más atributos, que dependen del tipo de mensaje.

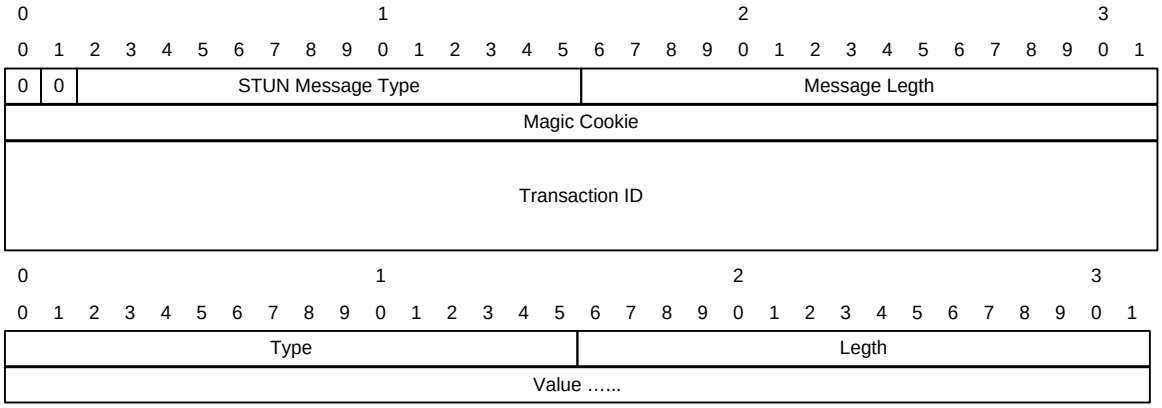

Figura 14. Estructura del mensaje STUN

3.2.2 Transacciones

STUN define dos tipos de transacciones, petición/respuesta e indicaciones.

Los clientes pueden realizar múltiples peticiones simultáneas con diferentes identificadores, sin necesidad de esperar por la terminación de una transacción para enviar la siguiente petición.

Sobre TCP, TCP es responsable de asegurar la entrega. Si el cliente no recibe una respuesta después de 7900 ms, considera que la transacción ha terminado sin respuesta (timeout).

Sobre UDP, el cliente retransmite las peticiones comenzando con un intervalo de RTO (Retransmision Time Out), multiplicado por dos para cada retrasmisión. Cuando se han alcanzado 7 intentos de retrasmisión, y si no se recibe respuesta después de 1.6

segundos desde la última retrasmisión, el cliente deberá considerar que la transacción ha fallado.

3.3 Diagrama lógico de NAT

El diagrama lógico del NAT utilizado para el desarrollo del proyecto es el que se muestra a continuación:

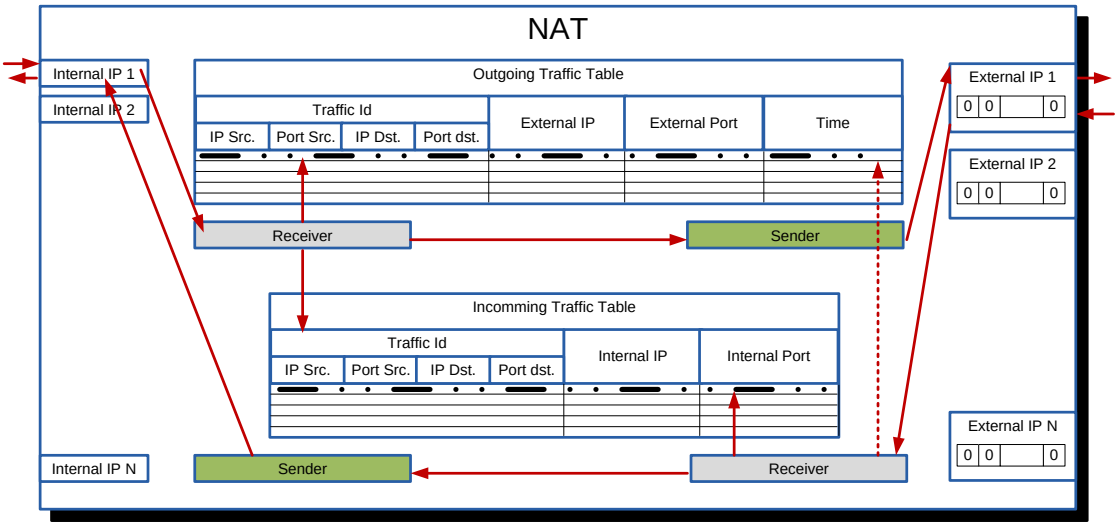

Figura 15. Diagrama simbólico de NAT

En este diagrama nos encontramos con varios elementos importantes que deberá implementar nuestro NAT. Estos elementos son:

Direcciones IP privadas: el NAT contendrá una o varias direcciones IP privadas para la conexión a una o varias redes privadas.

Direcciones IP y puertos públicos: el NAT deberá tener al menos una dirección pública pero podrá contar con un número ilimitado de ellas. La forma de asignar la dirección IP externa podrá ser pareada o aleatoria. Cada una de estas direcciones IP dispondrá de un mapa de bits de tamaño 2^{16} que se usará para asignar el puerto externo. En esta tabla cada posición se corresponde con un puerto, de forma que si la posición N contiene el valor 1, el puerto estará ocupado, mientras que si el valor es 0 el puerto estará libre. Esta tabla se usará para implementar la asignación de puertos con o sin sobrecarga y con o sin reserva.

Tabla de conversión de tráfico de salida: es una tabla de conversión de direcciones IP y puertos privados a direcciones IP y puertos públicos. A esta tabla la llamaremos **Outgoing Traffic.** Aunque la descripción de NAT expuesta en el párrafo anterior no es rigurosa, diremos también que estas asignaciones entre direcciones privadas y direcciones públicas se crean y se mantienen activas durante un tiempo T, que deberá ser configurable. Este tiempo determina el periodo máximo entre dos paquetes de salida

durante el cual el NAT considera que la comunicación no ha expirado. Por lo tanto la tabla deberá contener los siguientes campos:

- **Identificador de tráfico de salida**: es un identificador único compuesto por la dirección IP y el puerto del extremo privado y la dirección IP y el puerto del extremo público.

- **Dirección IP externa**: es la dirección pública del NAT que es asignada a la comunicación entre ambos extremos.

- **Puerto externo**: es el puerto que el NAT asigna a la comunicación entre ambos extremos.

- **Tiempo**: contendrá el momento de la última transición de tráfico. Este tiempo podrá contener el momento en el que se produjo la última comunicación de datos de salida, implementando así la funcionalidad descrita como refresco de salida en el capítulo 2 de este documento. Además deberá ser configurable para permitir el refresco de entrada y por lo tanto contendrá el momento de la última transición de paquetes de entrada o de salida.

Esta tabla de conversión de tráfico de salida permitirá la implementación de los diversos modos de funcionamiento de NAT en cuanto a la asignación de IP y puertos públicos en función de los extremos privado y público de la conexión (Ver apartado 2.3 Asignación de IP y Puerto). Según la topología NAT descrita, los modos de funcionamiento observados son:

- **Dependiente de la dirección y el puerto:** en este caso, el NAT genera una nueva entrada en la tabla de salida siempre que cambie cualquiera de las variables de la comunicación (IP y puerto privado o IP y puerto público). La tabla contendrá los cuatro elementos como clave única.

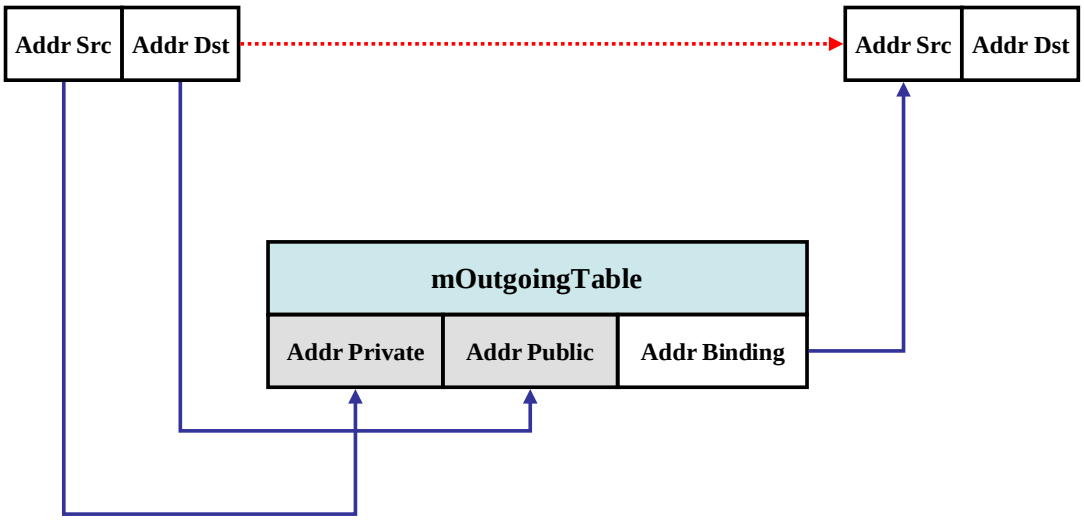

Figura 16. Tabla de tráfico de salida. NAT dependiente de la dirección y el puerto

- **Dependiente de la dirección:** en este caso, el NAT genera una nueva entrada en la tabla de salida siempre que cambie cualquiera de las variables de la comunicación a excepción del puerto de destino (IP y puerto privado o IP pública). En este caso la tabla contendrá estos tres elementos como clave única.

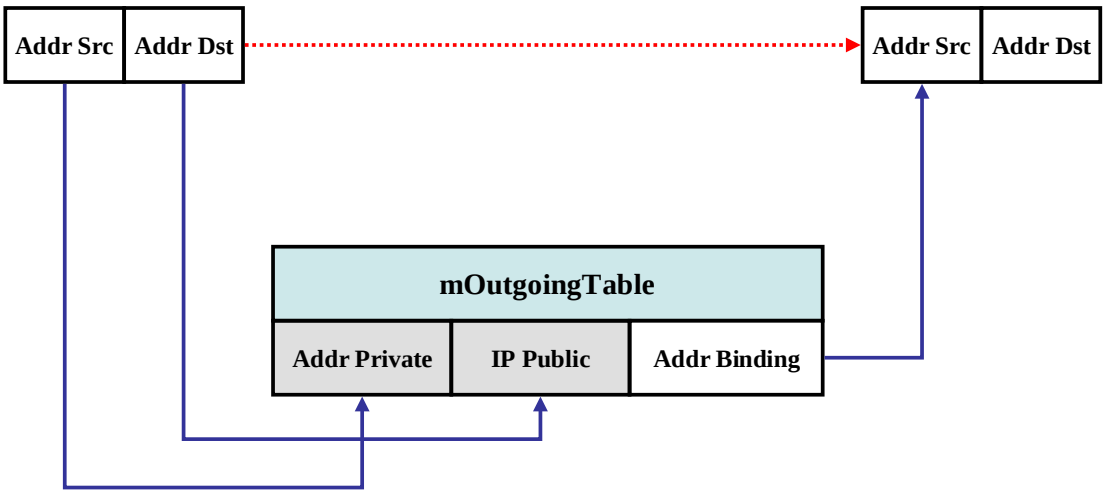

Figura 17. Tabla de tráfico de salida. NAT dependiente de la dirección.

- **Independiente del extremo:** en este caso, el NAT genera una nueva entrada en la tabla de salida siempre que cambie cualquiera de las variables privadas de la comunicación (IP y puerto privado). En este caso la tabla contendrá estos dos elementos como clave única.

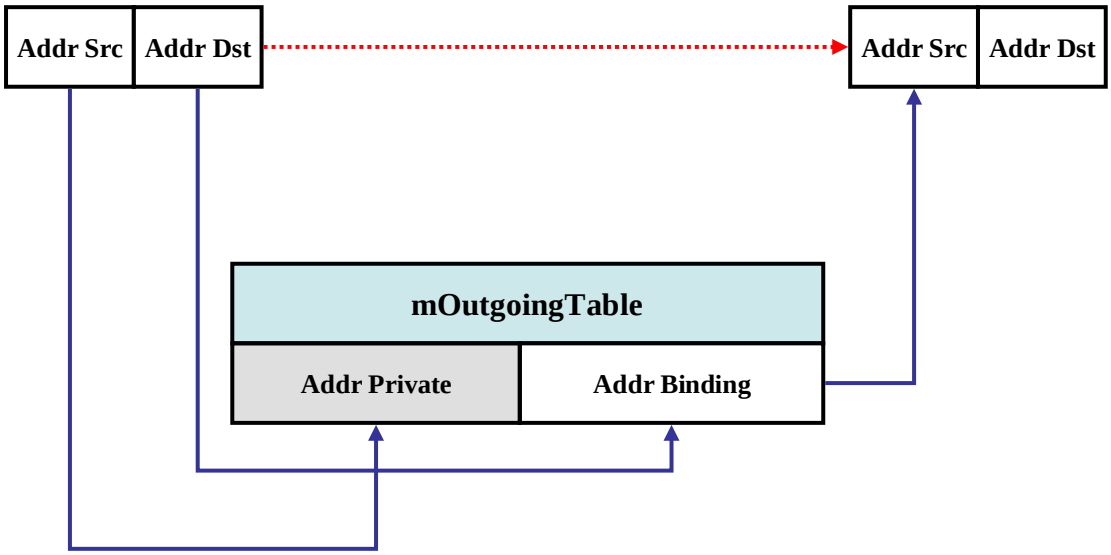

Figura 18. Tabla de tráfico de salida. NAT independiente del extremo.

Tabla de conversión de tráfico de entrada: si bien esta tabla no es estrictamente necesaria en la implementación, será de gran utilidad para agilizar la conversión de direcciones IP y puerto de los paquetes de entrada. Contendrá los siguientes campos:

- **Identificador de tráfico de entrada**: es un identificador único compuesto por la dirección IP y el puerto del extremo público y la dirección IP y el puerto asignados por el NAT.

- **Dirección IP privada**: es la dirección privada desde la que se inicio la comunicación.

- **Puerto privado**: es el puerto del extremo privado desde el que se inicio la comunicación.

Esta tabla de conversión de tráfico de entrada permitirá la implementación de los diversos modos de funcionamiento de NAT en cuanto al filtrado de paquetes de entrada (Ver apartado 2.8 Filtrado de paquetes de entrada).

- **Filtro dependiente de la dirección y el puerto:** en este modo de funcionamiento, el NAT resuelve la dirección privada de destino del paquete de entrada en función de los cuatro parámetros de la comunicación (IP y puerto del extremo público e IP y puerto asignados por el NAT).

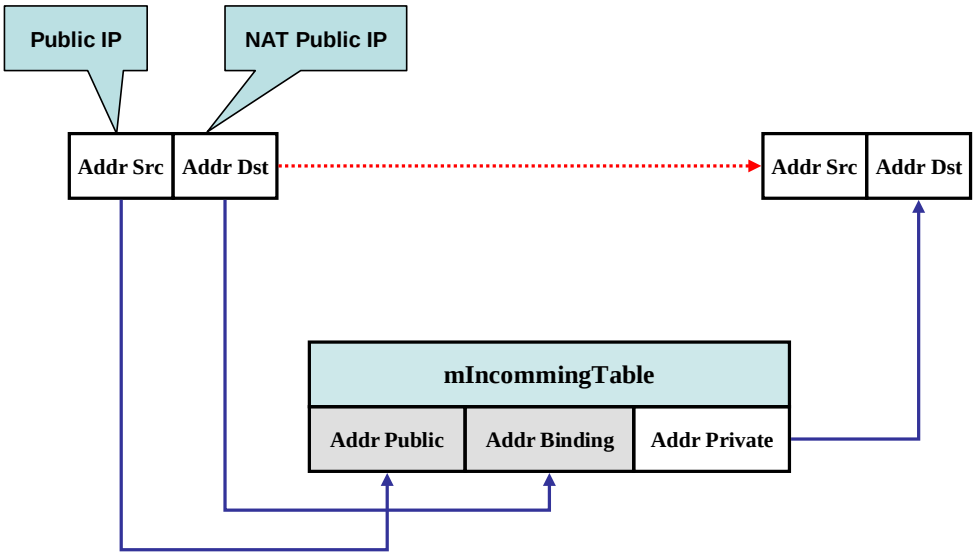

Figura 19. Tabla de tráfico de entrada. NAT dependiente de la dirección y del puerto.

Filtro dependiente de la dirección: en este modo de funcionamiento, el NAT resuelve la dirección privada de destino del paquete de entrada en función de tres parámetros de la comunicación (IP del extremo público e IP y puerto asignados por el NAT).

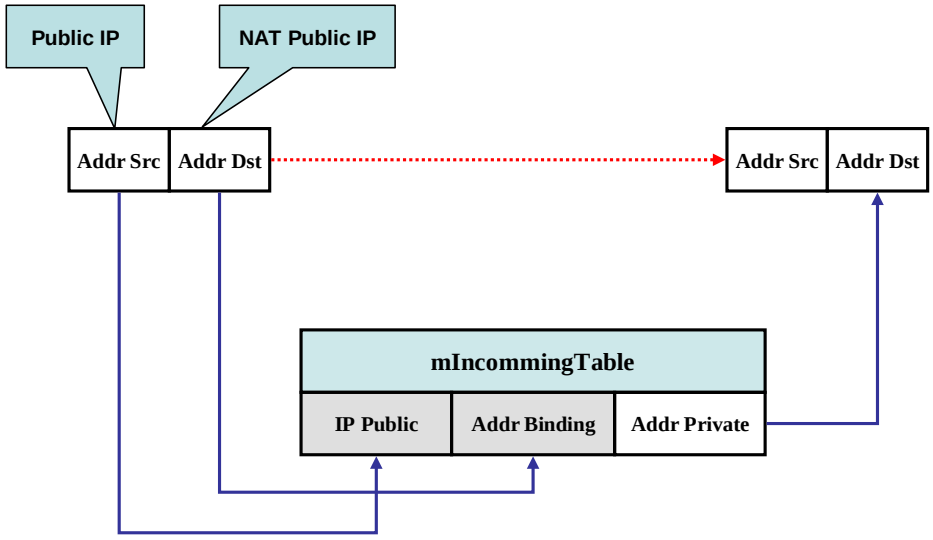

Figura 20. Tabla de tráfico de entrada. NAT dependiente de la dirección.

Filtro independiente de la dirección: en este modo de funcionamiento, el NAT resuelve la dirección privada de destino del paquete de entrada en función de la dirección IP y el puerto públicos asignados por el NAT.

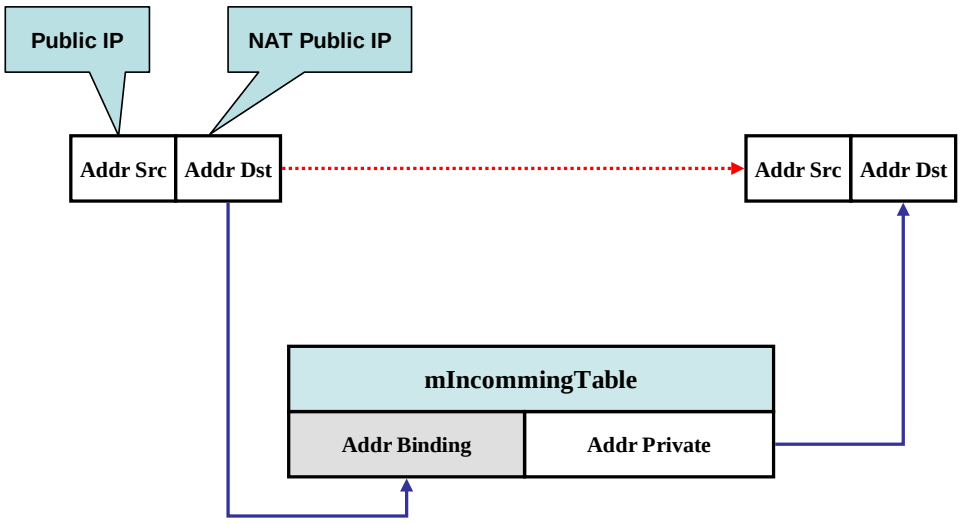

Figura 21. Tabla de tráfico de entrada. NAT independiente de la dirección.

3.4 Diagrama lógico de la solución

El escenario básico de la comunicación a través de NAT consta, como ya se ha mencionado anteriormente, de tres elementos:

- un cliente situado en una red privada,
- un NAT que cuenta con una o varias direcciones IP públicas,
- un host en la zona pública del NAT con el que se desea comunicar el cliente.

Gráficamente, el escenario más básico es el que se muestra en la siguiente figura:

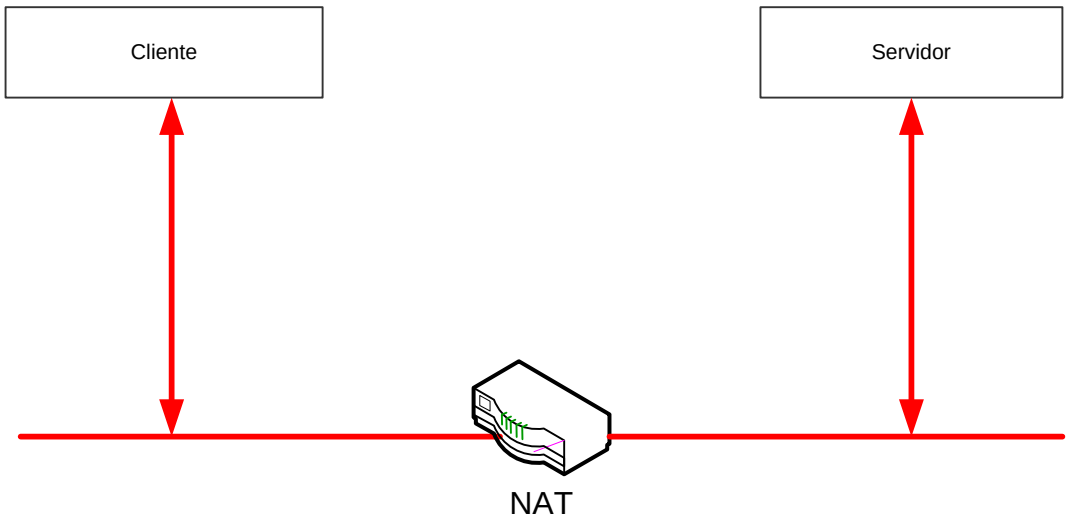

Figura 22. Escenario básico de conexión a través de NAT.

Partiendo de este escenario podemos hacer una primera división de grano grueso de los componentes software que vamos a necesitar en la que se incluye un cliente, un servidor y un dispositivo NAT.

Cada uno de estos elementos deberá ser capaz de enviar paquetes UDP con una dirección y puerto origen arbitrario. Estos paquetes deberán ser capturados por el siguiente elemento en la cadena de reenvíos de mensajes hasta alcanzar su destino. La secuencia de acciones necesarias para el envío de un mensaje desde el cliente hasta el servidor será por lo tanto la siguiente:

- El cliente con dirección 1 desea enviar un mensaje UDP al servidor con dirección 3, y crea un paquete con las direcciones origen y destino correspondientes.
- El NAT deberá capturar este paquete puesto que se encuentra en la misma red que el cliente y además el destino es alcanzable,
- El NAT deberá modificar el paquete recibido cambiando la dirección origen por una de sus direcciones públicas, por ejemplo la dirección 2. A continuación, almacenará en sus tablas internas la asociación de direcciones y reenviará el paquete modificado,
- Finalmente, el servidor capturará el paquete creado por el NAT, puesto que la dirección de destino coincide con la suya y el origen del paquete es el NAT. Es importante darse cuenta de que un elemento en la zona pública de un NAT no debe capturar todos los paquetes que le tengan a el como destino, sino que además debe de tener su origen en el NAT.

Por lo tanto, deducimos de esta secuencia de acciones que necesitaremos al menos un subcomponente capaz de enviar mensajes UDP cuya dirección y puerto de origen son

seleccionables, esto es, deberemos tener acceso a la cabecera del mensaje UDP para asignar el puerto y a la cabecera del mensaje TCP para asignar la dirección IP.

Además de un emisor necesitaremos un receptor de paquetes UDP. Para que la simulación sea lo más realista posible el receptor no deberá tener consciencia de que el emisor se encuentra en la misma máquina. Para ello deberá ser capaz de obtener los paquetes enviados por el emisor en la capa de nivel IP y filtrar aquellos que le sean de interés ignorando el resto. Para ello necesitaremos un subcomponente que aplicando un filtro de mensajes UDP, sea capaz de de escuchar en el nivel IP y seleccionar los paquetes que le interesen y ocultar al elemento de simulación el origen del mensaje. En este sentido la aplicación se ejecutará realizando un efecto ping-pong, en el que los paquetes enviados por un elemento rebotan hacia el elemento siguiente que se encuentra en la misma máquina y que cuenta con una sola dirección IP real. De esta forma conseguiremos simular tantas direcciones y puertos como sean necesarios.

A continuación se presenta un refinamiento del escenario básico en el que se observan los subcomponentes identificados en el proceso de análisis y sus interacciones:

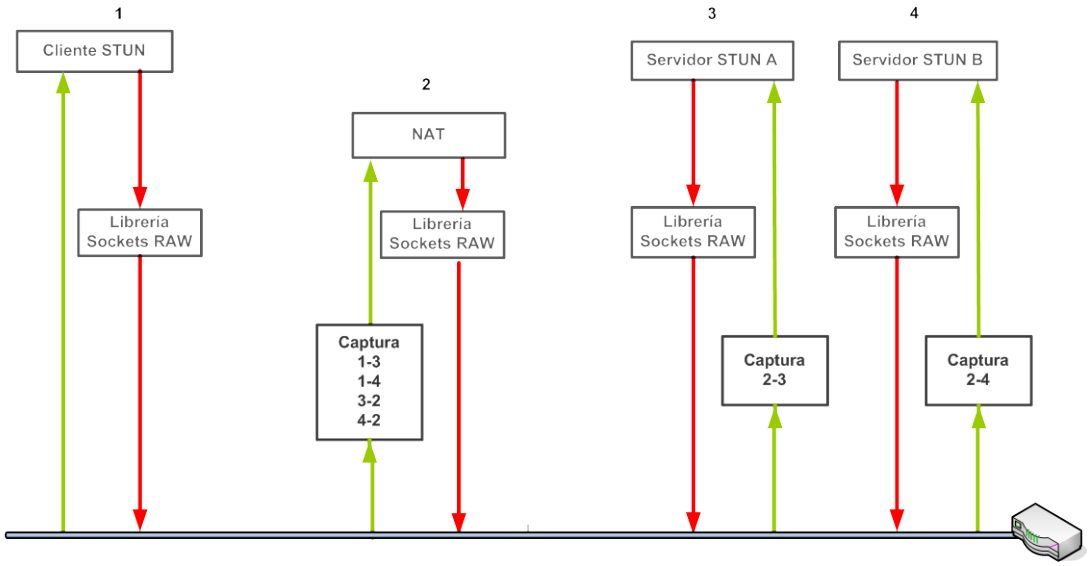

Figura 23. Descomposición del escenario básico de conexión a través de NAT

La solución propuesta para la observación del tráfico y el comportamiento del NAT podrá ser ampliada a escenarios más complejos como el que se presenta en la siguiente figura. En este caso, el cliente que se encuentra en la red privada esta conectado a una red externa a través de un NAT que a su vez accede a la red pública a través de un segundo NAT. Es este un escenario real puesto que algunos ISP utilizan esta técnica para lidiar con el problema de la escasez de direcciones IP y ofrecer servicios de acceso a Internet a un mayor número de usuarios del que en realidad podrían manejar en virtud del rango de direcciones IP del que disponen. Además en este caso, el cliente puede tener la necesidad tanto de acceder a un servidor que se encuentra en la red privada del operador para determinadas operaciones, como pueden ser de mantenimiento de páginas web o el acceso a servidores de correo, como de comunicarse con servidores que se encuentran en la red pública.

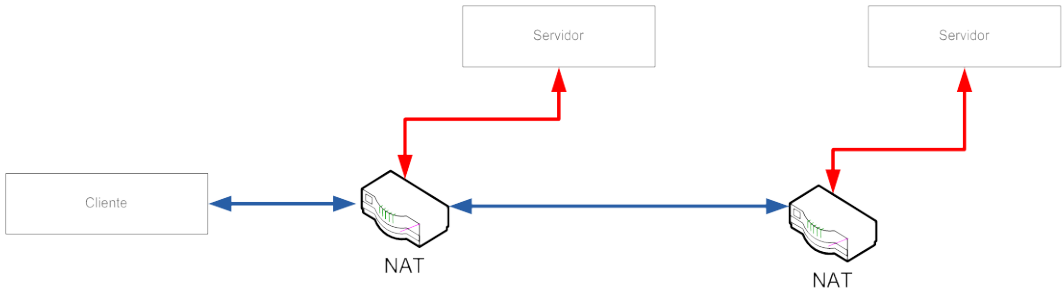

Figura 24. Escenario de conexión a través de NAT en cascada

Finalmente, se incluye la descomposición del segundo escenario de acuerdo a las subcomponentes propuestos para la consecución del objetivo de simulación de conexión a través de NAT:

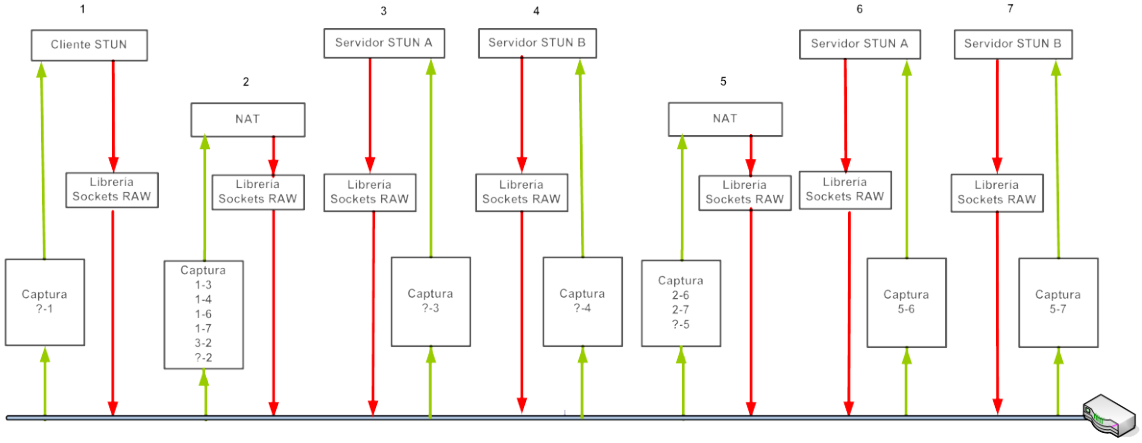

Figura 25. Escenario básico de conexión a través de NAT en cascada

Capítulo 4

Diseño

4.1 Introducción

En esta sección se muestra el diseño detallado de la solución.

El primer apartado introduce a muy alto nivel los componentes que vamos a necesitar y las interacciones necesarias para conseguir la simulación de comunicación a través de NAT.

El segundo apartado presenta la descomposición de la solución en librerías.

A continuación dedicaremos un apartado para cada librería en el que se describen las clases que la componen y sus responsabilidades.

Finalmente se presenta un diagrama detallado del diseño incluyendo todos los componentes trabajando en paralelo.

4.2 Objetivo del diseño

El objetivo principal del diseño es la creación de escenarios de prueba de la comunicación entre dos máquinas situadas en redes diferentes y que se comunican a través de un NAT, una de ellas actuando como cliente STUN y la otra como servidor STUN.

Para conseguir este objetivo, deberemos crear la ilusión de máquinas independientes. Para ello nos basaremos en la creación de hilos de ejecución simultánea simulando cada una de las partes. Utilizaremos un hilo por cada cliente, un hilo por cada servidor y dos hilos por cada NAT (uno para la captura y envió de paquetes desde la zona pública y otro para la captura y envío de paquetes desde la zona privada).

Cada uno de estos hilos estará compuesto por dos grandes bloques. El primer bloque se dedicará a la captura de aquellos paquetes UDP que le tengan como destino, en el caso de un extremo, o que deban transitar por el, en el caso de un NAT. El segundo bloque se dedicará a enviar aquellos paquetes que consideré oportunos como peticiones o respuestas si se trata de un extremo o reenvíos de paquetes modificados si se trata de un NAT.

Disponemos de varías utilidades que nos permitirán conseguir esta ilusión de máquinas remotas. Para la captura de paquetes utilizaremos la librería pcap que es una librería de código abierto para la captura de paquetes y análisis de red. Para el envío de paquetes utilizaremos los sockets de tipo RAW que permiten la manipulación de las cabeceras IP y UDP antes de inyectarlos en la red.

El modelo finalmente buscado es el siguiente:

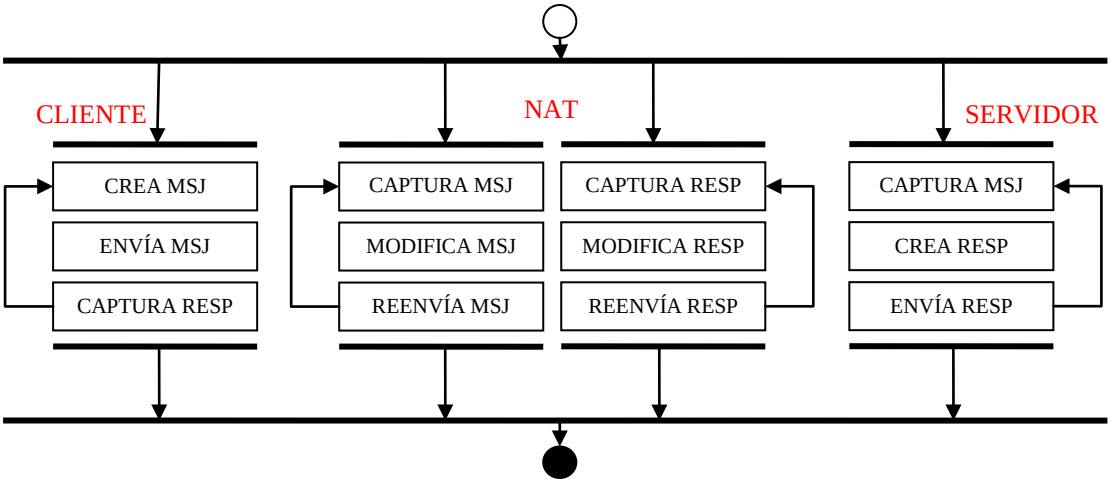

Figura 26. Diseño de alto nivel

4.3 Organización de las clases en librerías

Las librerías se han desarrollado en C++ con el objetivo de que sean portables entre los dos sistemas operativos más extensamente utilizados en la actualidad, esto es Unix y Windows.

Bajo este principio de portabilidad, se puede hacer una primera división de las librerías en dos grupos bien diferenciados. El primer grupo comprende el núcleo de la solución y es totalmente portable. El segundo grupo, que es en realidad una única librería (Simulator), contiene el interface de presentación de datos e interacción con el usuario. En esta librería no se ha seguido el principio de portabilidad y se ha decidido implementarlo en Windows con la librería de clases MFC (Microsoft Foundation Classes).

Gráficamente, las librerías y sus dependencias se muestran en la siguiente figura:

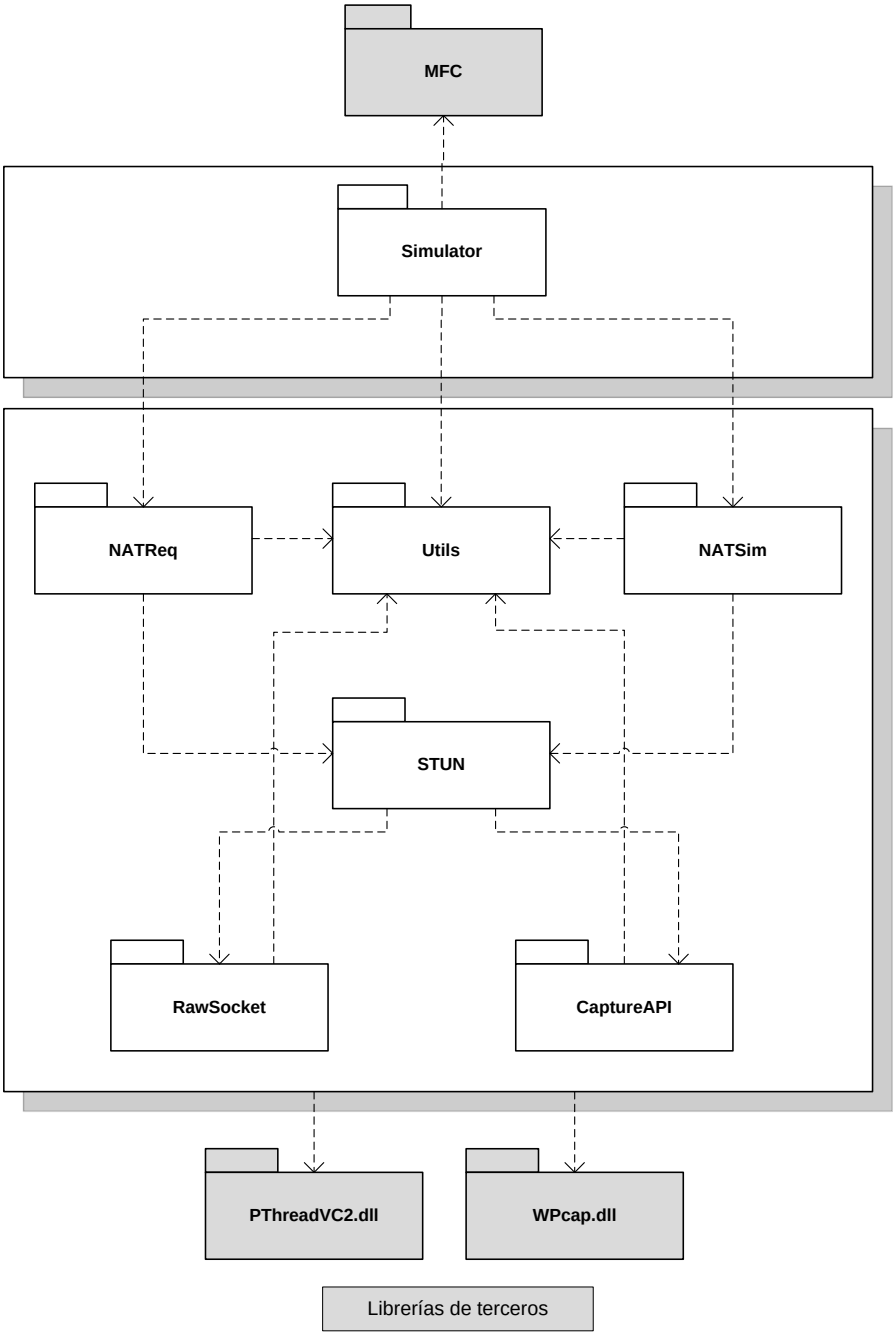

Figura 27. Descomposición en librerías de la solución

La funcionalidad contenida en cada una de las librerías se describe en los apartados siguientes.

4.3.1 Librería de Utilidades (Utils)

Se trata de una librería de utilidades que implementa las funcionalidades básicas requeridas por el resto de la aplicación. Contiene las siguientes clases:

CUtilByteArray: implementa una cadena de bytes de longitud variable. Facilita la codificación y decodificación de los mensajes.

CUtilConfigFile: implementa un fichero de configuración para la lectura y escritura de parámetros.

CUtilCriticalSection: encapsula una región crítica basada en **mutex**. Facilita la portabilidad del código de la aplicación.

CUtilString: implementa una cadena de caracteres de longitud variable.

CUtilTrace: ofrece un fichero de traza para el seguimiento y la depuración de la actividad de la aplicación.

CQueue: implementa una cola de objetos de cualquier tipo.

CArray: implementa un array de objetos de cualquier tipo.

CIdTbl: implementa un array ordenado de objetos de cualquier tipo para cualquier clase de índice de ordenación.

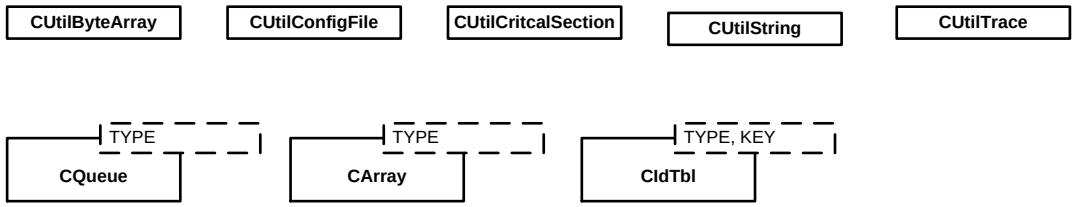

Figura 28. Diagrama de clases de la librería de utilidades

4.3.2 Librería de captura de paquetes (CaptureAPI)

La librería PCap (Libpcap para Unix y WinPcap para Windows) puede ser utilizada para capturar los paquetes que viajan por toda la red y, en sus versiones más recientes, para transmitir los paquetes en la capa de enlace de una red, así como para conseguir una lista de los interfaces de red que se pueden utilizar.

Para el propósito del proyecto, se ha utilizado la capacidad de captura de paquetes de la librería PCap, no así la capacidad de envió de paquetes, para lo cual se han utilizado los sockets de tipo RAW, como se describe en la siguiente sección.

La librería CapturaAPI permite, basándose en PCap, la captura de paquetes aplicando un filtro configurable durante la construcción del objeto. El filtro admite un número indefinido de direcciones IP y puertos del origen del mensaje así como un número indefinido de direcciones IP y puertos del destino.

La creación de un objeto de esta clase implica la creación de un nuevo hilo de ejecución que captura los paquetes y los inserta en una cola de recepción de la que se pueden extraer todos los mensajes obtenidos de la red. Funciona como un productor de mensajes para las capas superiores de la aplicación.

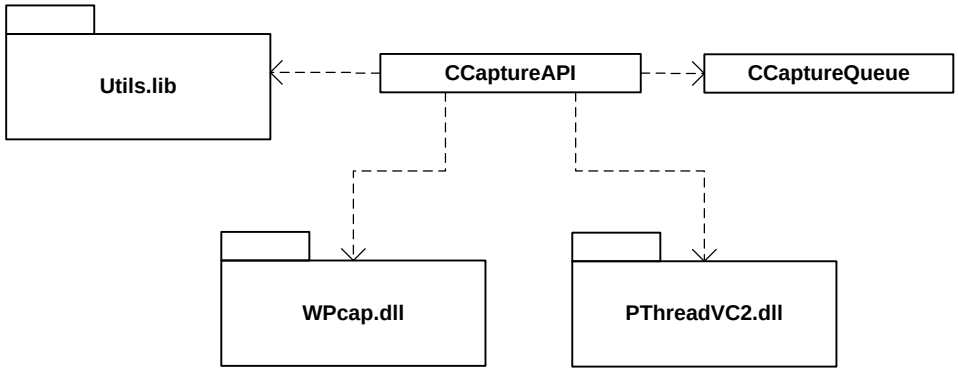

Figura 29. Diagrama de clases de la librería de captura de paquetes

4.3.3 Librería de sockets Raw (RAWSocket)

Permite el acceso a la cabecera IP durante la construcción y el envío de mensajes. Por medio de esta librería conseguiremos inyectar mensajes en la red creando la cabecera IP con las direcciones origen y destino que deseemos, y conseguiremos que el kernel respete nuestra cabecera sin sobrescribirla. Esta característica de los sockets RAW se ha utilizada en muchas ocasiones para provocar ataques de denegación de servicio (DoS attack) y por esta razón algunas versiones del sistema operativo Windows, concretamente Windows XP con SP2 o posterior y Windows Vista, han limitado su uso. En cualquier caso, el uso de este tipo de sockets requiere privilegios de administración para su uso.

El siguiente esquema muestra los diferentes tipos de sockets del dominio AFI_INET, y en ella se observa como los sockets de tipo RAW nos permitirán incluir una cabecera TCP o UDP con las opciones que deseemos. Esta característica de los sockets RAW se utilizará junto con la opción del socket IP_HDRINCL, que permite indicar que la aplicación proporciona la cabecera IP.

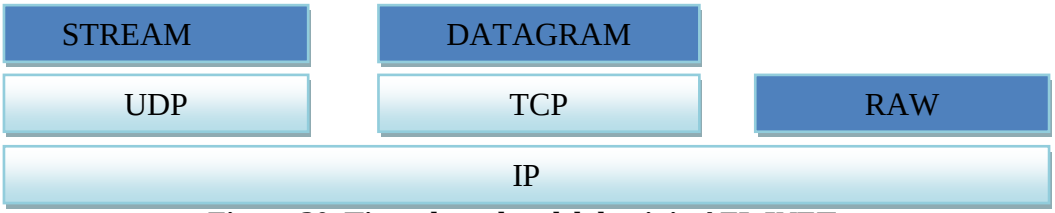

Figura 30. Tipos de socket del dominio AFI_INET

Esta librería contiene una única clase de nombre CRawSocket que implementa la siguiente interfaz:

bool CreateSender()

Crea un socket de tipo RAW y activa la opción de creación de cabecera IP.

```
mHandle = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
setsockopt (mHandle, IPPROTO_IP, IP_HDRINCL, &on, sizeof (on));
```

bool Send(CUtilByteArray & data,
const char * saddr, int sport,
const char * daddr, int dport);

Crea el datagrama UDP y lo envía con la IP y puerto origen hacía la IP y puerto destino.

```
datagram = new byte[dataLength + sizeof(struct ip) + sizeof(struct udphdr)];

memset(&iph, 0, sizeof(struct ip));
memset(&udph, 0, sizeof(struct udphdr));

iph.ip_v = 0x04;
iph.ip_hl = 0x05;
iph.ip_tos = 0x00;
iph.ip_len = sizeof (struct ip) + sizeof (struct udphdr) + dataLength;
iph.ip_id =  4321;
iph.ip_off = 0x00;
iph.ip_ttl = 0x64;
iph.ip_p = IPPROTO_UDP;
iph.ip_sum = 0x0;
iph.ip_src.s_addr = inet_addr (saddr);
iph.ip_dst.s_addr = inet_addr (daddr);
memcpy(datagram, &iph, sizeof(struct ip));

udph.uh_sport = htons(sport);
udph.uh_dport = htons(dport);
udph.uh_ulen = htons((word)(sizeof(struct udphdr) + dataLength));
udph.uh_sum = 0;
memcpy(datagram +  sizeof(struct ip), &udph, sizeof (struct udphdr));

memcpy(datagram +  sizeof (struct ip) + sizeof (struct udphdr), data, dataLength);

memset(&sin, 0, sizeof(sin));
sin.sin_family = AF_INET;
sin.sin_port = htons (dport);
sin.sin_addr.s_addr = inet_addr(daddr);

sendto(mHandle,
```

```
            (const char *)datagram,
            iph.ip_len,
            0,
            (struct sockaddr *) &sin,
            sizeof (struct sockaddr));

    delete datagram;
```

bool TerminateSender();

Cierra el socket de envío.

close(mHandle)

4.3.4 Librería STUN (STUN)

Implementa la funcionalidad básica del protocolo STUN. Proporciona además un servidor y un cliente ligero de STUN. El siguiente diagrama ilustra las relaciones entre las clases:.

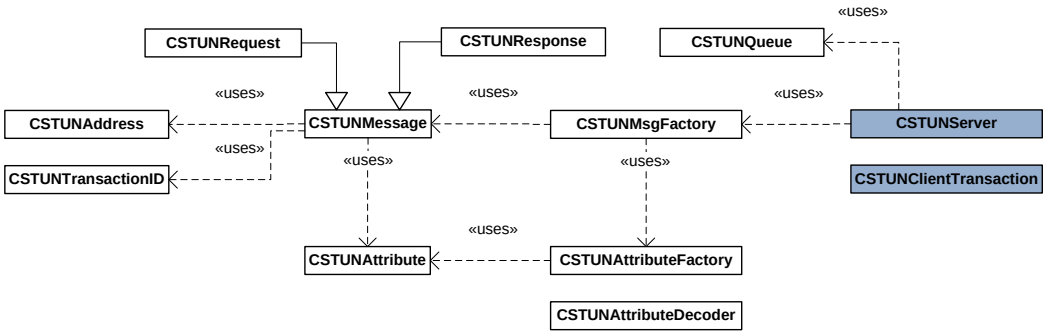

Figura 31. Diagrama de clases de la librería STUN

Las responsabilidades de las clases se describen a continuación:

CSTUNAddress: implementa una dirección STUN como una pareja IP y puerto.

CSTUNTransactionID: ofrece un identificador de transacción de un mensaje STUN. El cliente STUN inserta un identificador de transacción en las peticiones que deberá ser el mismo que reciba en la respuesta del servidor.

CSTUNQueue: representa una cola de mensajes STUN.

CSTUNMessage: encapsula un mensaje STUN.

CSTUNRequest: es una derivación de CSTUNMessage para mensajes de tipo petición.

CSTUNResponse: es una derivación de CSTUNMessage para mensajes de tipo respuesta.

CSTUNAttribute: implementa los distintos tipos de atributos que puede contener un mensaje STUN.

CSTUNMsgFactory: clase factoría de mensajes STUN.

CSTUNAttributeFactory: clase factoría de atributos STUN.

CSTUNAttributeDecoder: codifica y decodifica los atributos contenidos en un mensaje de tipo STUN.

CSTUNClientTransaction: representa una transacción de un cliente STUN. Envía y espera recibir la respuesta desde el servidor. Si la transacción no se completa con éxito, reintenta un número de veces configurable. Permite enviar el mensaje UDP desde una dirección IP que no es necesariamente ninguna de las IP asignadas a los interfaces de la máquina, apoyándose en las clases CRawSocket y CCaptureAPI.

CSTUNServer: implementa un servidor STUN. Crea un hilo de ejecución que representan dos máquinas independientes con distintas IP basándose en las clases CRawSocket y CCaptureAPI. Lee su configuración apoyándose en CUtilConfigFile.

Dada la importancia de las clases CSTUNClientTransaction y CSTUNServer, en los siguientes apartados se muestra con mayor detalle la estructura de clases que colaboran con estas para conseguir la funcionalidad STUN necesaria para el desarrollo del proyecto.

4.3.4.1 Cliente STUN

El cliente STUN implementa la petición de mapeo (**BINDING REQUEST**), permitiendo seleccionar las direcciones IP y puerto de origen y destino con los que se creará el paquete UDP que será inyectado en la red. Este paquete, podrá contener direcciones validas de cliente y servidor, con lo que el cliente se comportará en modo real, o bien podrá contener direcciones ficticias, que permitirán realizar una simulación completa dentro de la misma máquina.

Para conseguir estos dos objetivos, el cliente STUN se compone de una instancia de la clase CRawSocket, que le permitirá inyectar paquetes con cualquier dirección y puerto de origen o destino y de una instancia de la clase CCaptureAPI, que como se vio en el apartado referente a la librería de captura, permite capturar paquetes definiendo un filtro por IP y puerto origen y destino. De esta manera, un cliente puede enviar mensajes desde la dirección d1 y puerto p1, y capturar la respuesta creando un filtro de recepción de paquetes para esta combinación IP y puerto.

El cliente crea el mensaje STUN de tipo BINDING REQUEST y le solicita a la clase CRawSocket que lo envíe a una dirección y puerto destino, la del servidor STUN, con una dirección IP y puerto de origen seleccionable. A continuación espera recibir en su respectivo hilo de captura de la clase CCaptureAPI, un paquete de respuesta a su petición en un tiempo inferior a T. Si en este tiempo no recibe la respuesta reintenta la transacción enviando hasta N veces el mismo paquete. Si no recibe repuesta da por terminada la transacción. Si por el contrario recibe una respuesta, devuelve el cuerpo del mensaje UDP a la capa superior, la instancia e la clase CSTUNClientTransaction.

Los siguientes gráficos muestran la estructura de clases simplificada y un diagrama de secuencia del cliente STUN.

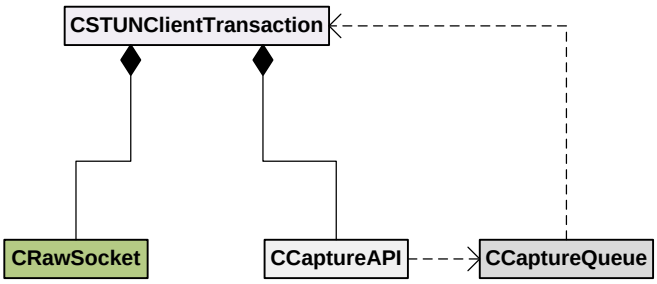

Figura 32. Diagrama de clases de Cliente STUN

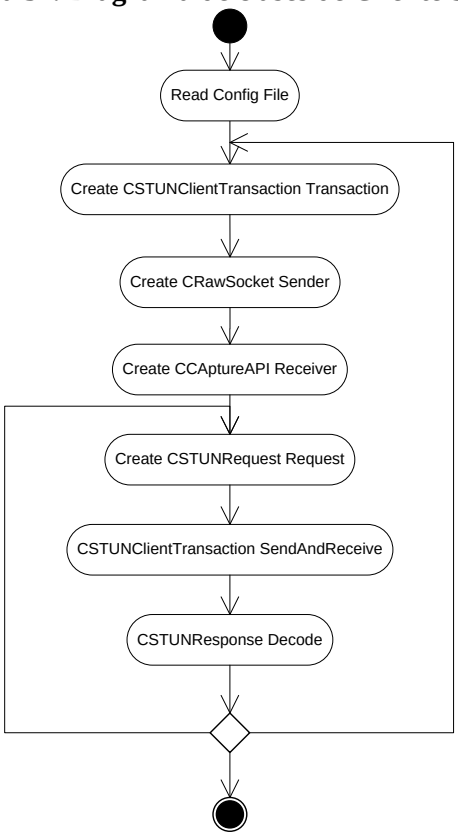

Figura 33. Diagrama de secuencia de Cliente STUN

4.3.4.2 Servidor STUN

El servidor STUN implementa la respuesta a la petición de BINDING REQUEST. Esta petición puede contener un atributo llamado CHANGE REQUEST. Este atributo indica al servidor que el cliente desea que la respuesta sea enviada cambiando la dirección IP (CHANGE IP) y/o el puerto (CHANGE PORT) origen de la respuesta por la de un servidor STUN secundario. Esto es, un cliente envía una petición de mapeo hacia el servidor STUN A cuya dirección es IpA y el puerto PortA. Si la petición contiene activada la bandera CHANGE IP, el paquete IP de respuesta contendrá como dirección origen IpB distinta de IpA. Si la petición contiene activado el flag CHANGE PORT, el paquete UDP de respuesta contendrá como puerto origen PortB que es diferente del puerto PortA. Ambos flags pueden estar activados o desactivados y son independientes. En cualquier caso, si el servidor dispone de una dirección y puerto alternativos, debe incluir en la respuesta a la petición un atributo OTHER_ADDRESS con esta información.

La siguiente tabla ilustra el comportamiento del servidor STUN con dos direcciones IP y puertos alternativos. En esta tabla se representa cualquiera de las dos direcciones IP y puertos de destino de la petición como Da y Dp (Destination Address y Destination Port), y la dirección alternativa como Ca y Cp (Changed Address y Changed Port):

Banderas	IP Origen	Puerto Origen	OTHER-ADDRESS
None	Da	Dp	Ca:Cp
Change IP	Ca	Dp	Ca:Cp
Change port	Da	Cp	Ca:Cp
Change IP and port	Ca	Cp	Ca:Cp

Tabla 8. Impacto de las banderas en la petición y OTHER-ADDRESS en respuesta.

Las siguientes gráficas muestran las clases que componen el servidor STUN y el diagrama de secuencia del mismo.

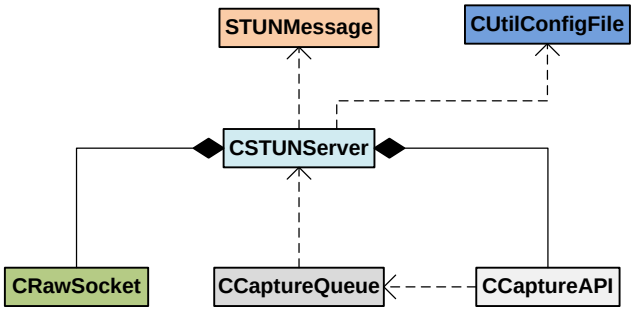

Figura 34. Diagrama de clases de Servidor STUN

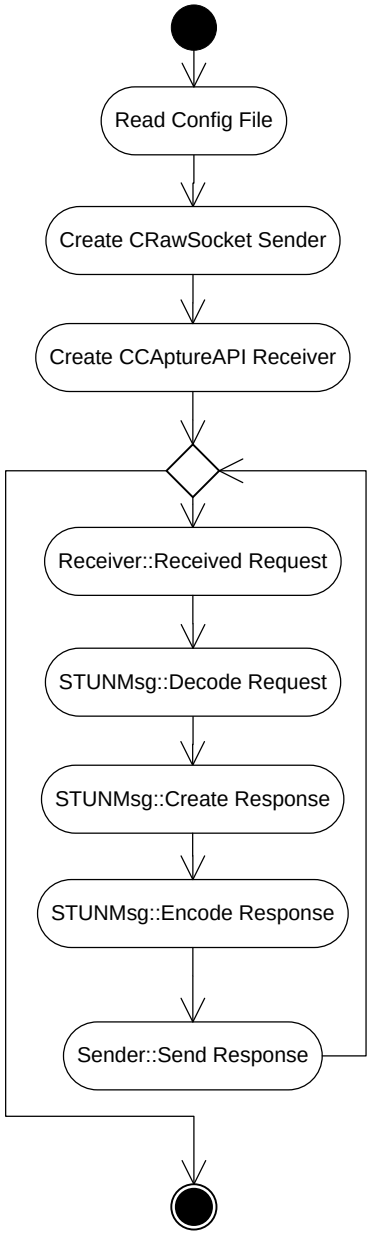

Figura 35. Diagrama de secuencia de Servidor STUN

4.3.5 Simulador NAT (NATSim)

El simulador del NAT es la pieza fundamental para terminar de completar la simulación. Hasta este momento hemos mostrado como podemos implementar un cliente y un servidor STUN sobre UDP que apoyados en las clases de captura y envío de paquetes podrían funcionar en el mismo host, bajo diferentes direcciones IP. Para completar la simulación, necesitamos algún tipo de componente que sea capaz de interceptar el tráfico, modificar el contenido de las cabeceras IP y UDP, y reenviar los paquetes modificados. Este componente es lógicamente el dispositivo NAT.

Un NAT real dispondrá de al menos dos interfaces de red, uno dedicado a la zona privada, con una dirección IP en el rango de direcciones de la red privada, y otro dedicado a la red pública, con una dirección IP en el rango de direcciones públicas. En

nuestra simulación conseguiremos esta diferenciación de interfaces creando dos hilos de ejecución independientes. Cada uno de estos hilos dispondrá de un componente para capturar y otro para enviar paquetes UDP. Es obvio que el NAT no debe tener conocimiento del tipo de tráfico que transita por el y por lo tanto no es necesaria ninguna capa tipo STUN para la implementación del NAT. Es cierto que existen NAT que proporcionan funcionalidad añadida a niveles superiores del nivel TCP o UDP, pero en nuestra implementación solo actuaremos sobre los paquetes en este nivel.

Las clases que necesitaremos se muestran en la siguiente figura:

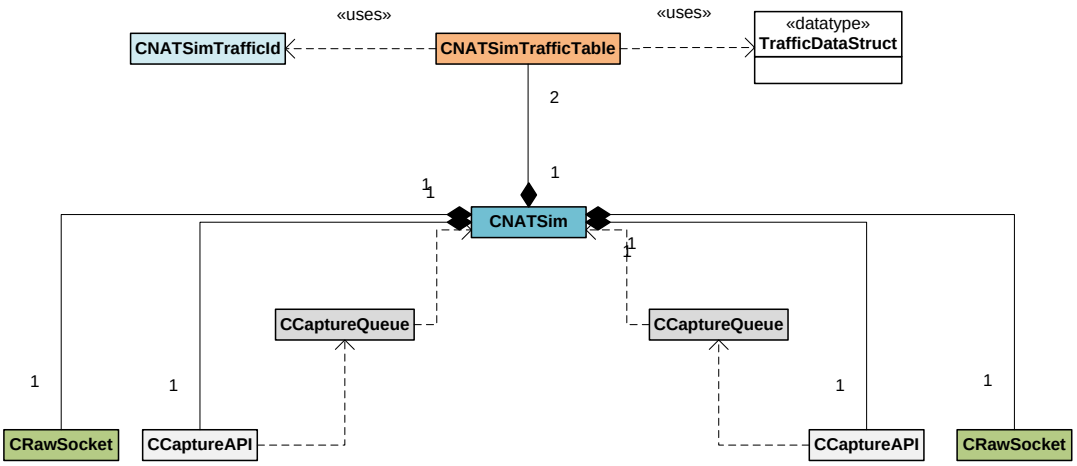

Figura 36. Diagrama de clases de la librería NATSim

Las responsabilidades de cada clase son las siguientes:

CNATSimTrafficId: representa un identificador único para el trafico de entrada o de salida del NAT. Está compuesto por una dirección IP y un puerto origen y una dirección IP y un puerto destino. Definiremos una relación de orden para este tipo de clave de forma que las tablas del NAT basadas en esta clave puedan estar ordenadas. Esta relación de orden es la siguiente:

Sean A y B dos claves del tipo CNATSimTrafficId.
Sean IPsA e IPsB, las direcciones IP origen de las claves A y B respectivamente.
Sean PsA y PsB los puertos origen de las claves A y B respectivamente.
Sean IPdA e IPdB, las direcciones IP destino de las claves A y B respectivamente.
Sean PdA y PdB los puertos destino de las claves A y B respectivamente.

A > B si y solo si:

IPsA > IPsB o
IPsA = IPsB y PsA > PsB o
IPsA = IPsB y PsA = PsB y IPdA > IPdB o
IPsA = IPsB y PsA = PsB y IPdA = IPdB y PdA > PdB

A = B si y solo si:

IPsA = IPsB y PsA = PsB y IPdA = IPdB y PdA = PdB

A < B si y solo si:

IPsA < IPsB o
IPsA = IPsB y PsA < PsB o
IPsA = IPsB y PsA = PsB y IPdA < IPdB o
IPsA = IPsB y PsA = PsB y IPdA = IPdB y PdA < PdB

TrafficDataStruct: representa una entrada en la tabla de tráfico. Contiene una clave de tipo CNATSiTrafficId, una dirección IP, un puerto y el momento de la última transición de tráfico.

CNATSimTrafficTable: representa una tabla del NAT para almacenar los mapeos o asociaciones de direcciones IP privadas a direcciones IP públicas, y su correspondiente dirección IP y puerto públicos del NAT asignados a la comunicación. Un simulador de NAT contendrá dos tablas de tráfico. La primera corresponde a las asociaciones de tráfico de salida y la segunda corresponde a las asociaciones de tráfico de entrada. La primera tabla se alimentará de los paquetes de salida, creando nuevas asociaciones cuando sea necesario en función del tipo de NAT, tal y como se comentó en el apartado 3.3. Se implementará como una subclase de la clase CIdTbl, que recordemos que implementa una tabla de elementos ordenados por clave única, en la que los elementos almacenados son del tipo TrafficDataStruct y la clave de ordenación es del tipo CNATSimTrafficId.

CNATSim: representa el simulador de NAT. Esta clase se crea como una agregación de cuatro subcomponentes dedicados a la simulación de los interfaces público y privado y de las tablas de control del tráfico de entrada y de tráfico de salida. Adicionalmente, para la lectura de la configuración del NAT, se utilizará una instancia de la clase CUtilConfigFile para la inicialización de los parámetros que definirán el comportamiento del NAT. El NAT contará con parámetros para:

- el tipo de mapeo en función del los extremos de la conexión,
- el tipo de filtrado de paquetes de entrada,
- el tiempo de conservación del mapeo en ausencia de tráfico,
- el tipo de refresco del mapeo como de entrada y salida o solo de salida,
- la conservación de puertos,
- la sobrecarga de puertos,
- el modo de asignación de direcciones IP (pareado o aleatorio).

Para completar la funcionalidad buscada, el simulador del NAT contendrá además una lista de direcciones IP públicas y para cada IP pública contendrá además un elemento de la clase CNATSimPortsArray. Esta clase será implementada como una subclase de CIdTbl con elementos del tipo PortDataStruct ordenados por dirección IP. Los elementos PortDataStruct contendrán a su vez una dirección IP y un vector de bits para representar la disponibilidad de los puertos para cada dirección IP pública del NAT.

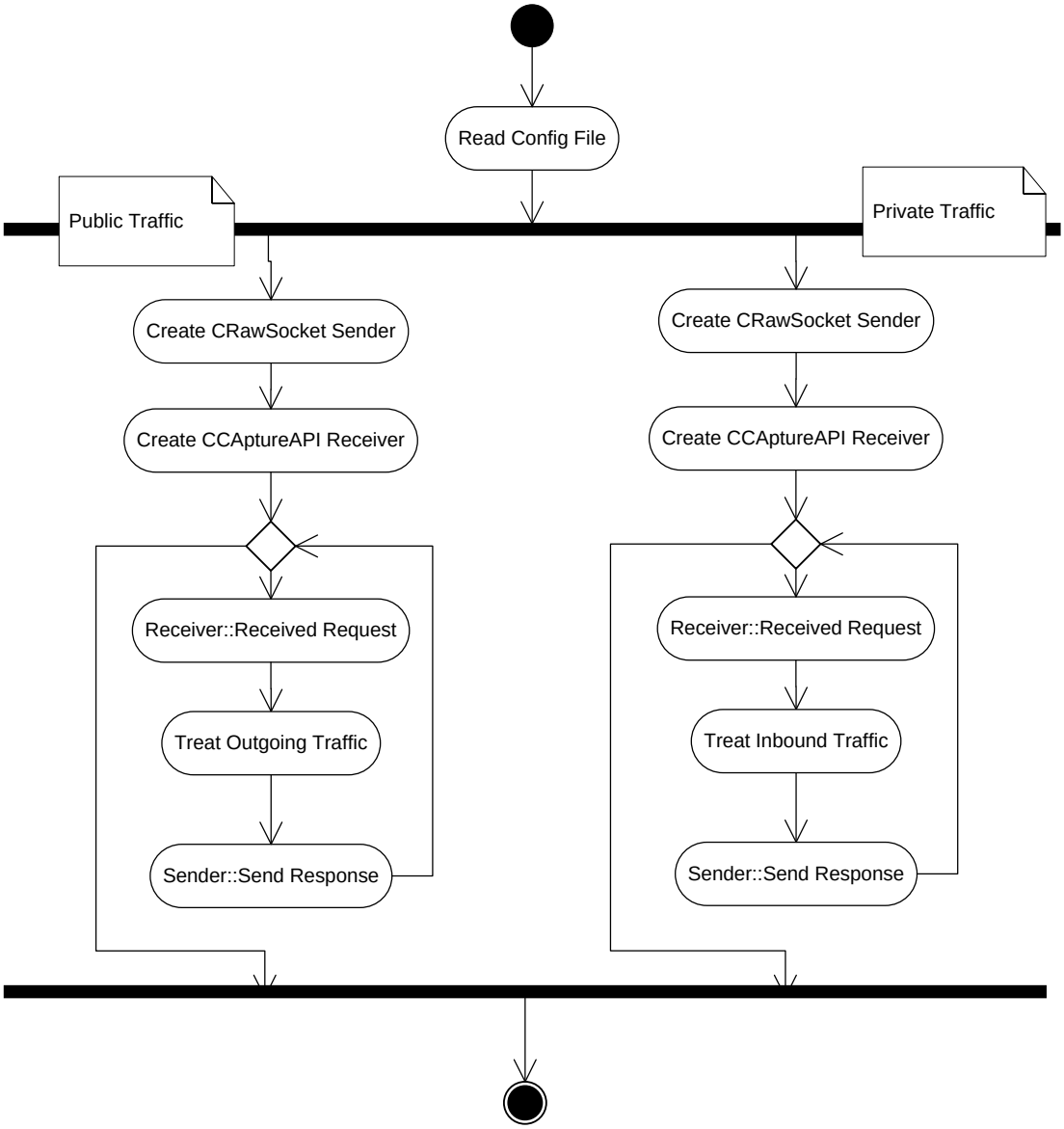

Figura 37. Diagrama de secuencia del simulador de NAT

4.3.6 Requisitos NAT (NATReq)

La librería de requisitos NAT contiene clases de implementación de los requisitos establecidos en el apartado *2.15 Requisitos NAT* de este documento.

La nomenclatura seguida para determinar los nombres de las clases de pruebas e identificar el requisito que implementan es muy sencilla. Cada clase recibe el nombre CNATReq**N**, donde N es el número de requisito implementado.

Esta librería utiliza el cliente ligero STUN implementado en la clase CSTUNClientTransaction de la librería STUN. Asimismo usa la clase CUtilConfigFile de la librería Utils para la lectura de los parámetros necesarios en la ejecución del test.

Asume que la configuración de los clientes STUN se encuentra contenido en el fichero STUNClients.config, conteniendo bloques de información que representan a cada uno de los clientes STUN presentes en el escenario. Como ejemplo se muestra uno de estos bloques de información de uno de los clientes STUN, en el que se guardan los datos relativos al cliente **A**:

```
#STUNClient
ClientADev=\Device\NPF_{9C9228F1-34FA-4517-9BF0-C6A85F775E53}
ClientAIP=192.168.3.2
ClientAPort=10000
```

Asume también que la configuración del servidor STUN se encuentra alojada en el fichero de configuración con nombre STUNServer**X**.config, donde X es una cadena de caracteres que permite identificar el servidor STUN que se utilizará en el test.

Como se verá más adelante en este documento, la actividad de la clase queda reflejada en un fichero de traza basado en la clase CUtilTrace de la librería Utils.

Todas las clases de esta librería ofrecen una interfaz muy similar a la que se muestra a continuación:

bool Test(CUtilString & fromClient, CUtilString & toServer, CUtilString & report);

La ejecución del test devuelve un valor booleano indicando si el test se ha realizado sin ningún error y por referencia una cadena de caracteres que contiene el resultado del test en cuestión. Como parámetros de entrada recibe el identificador del cliente y del servidor STUN a utilizar.

Como ejemplo de uso se muestra en pseudo código la implementación de una transacción STUN y la interpretación de los resultados.

```
// Crea los lectores de ficheros de configuración
CUtilConfigFile ConfigClient("STUNClients.config");
CUtilConfigFile ConfigServer("StunServer" + toServer + ".config");

// Lee parámetros
CUtilString sourceAddress=ConfigClient.GetString("STUNClient","Client"+fromClient+"IP");

word sourcePort = (word)ConfigClient.GetLong("STUNClient", "Client" + fromClient + "Port");
CUtilString device = ConfigClient.GetString("STUNClient", "Client" + fromClient + "Dev");

CUtilString addressSTUNServer = ConfigServer.GetString("STUNServer", "ServerAIP");
word portSTUNServer = ConfigServer.GetLong("STUNServer", "ServerPort1");

Trace("**DEBUG**", __FUNCTION__ , "Starting mapping behavior test...");

// Crea transación STUN
CSTUNClientTransaction STUNTransaction;
STUNTransaction.Create(sourceAddress, sourcePort, device, ConfigServer);

CSTUNTransactionID          TransactionID;
CSTUNAddress                STUNAddress;

// Crea contenedores de mensajes (envio y recepción)
CUtilByteArray      msgToSend;
CUtilByteArray      *msgReceived;

// Test (mensaje de descubrimiento con flags desactivados
CSTUNRequest & STUNRequest = CSTUNMsgFactory::createBindingRequest(false, false);


CSTUNAttribute      changedAddress;
```

```
CSTUNAttribute      xorMappedAddress;
CSTUNAddress        xorMappedAddressCalculated;
CSTUNAttribute      mappedAddress;

CSTUNTransactionID transactionID;

// Codifica los mensajes
STUNRequest.encode(msgToSend);
STUNRequest.Trace();

// Envía petición y recibe respuesta
STUNTransaction.SendAndReceive(
                                msgToSend,
                                &msgReceived,
                                sourceAddress,
                                sourcePort,
                                addressSTUNServer,
                                portSTUNServer );

// Decodifica y traza la respuesta
CSTUNResponse & STUNResponse=CSTUNMsg::decode( msgReceived);
STUNResponse.Trace();

// Obtiene identificador de transacción y attributos de la respuesta
transactionID = STUNResponse.getTransactionID();

STUNResponse.getAttribute(CHANGED_ADDRESS, changedAddress);
STUNResponse.getAttribute(XOR_MAPPED_ADDRESS,xorMappedAddress);
STUNResponse.getAttribute(MAPPED_ADDRESS, mappedAddress) ;

// Interpreta los resultados
if ( mappedAddress.getAddress() == sourceAddress && mappedAddress.getPort() == sourcePort )
{
      Trace("**INFO **", __FUNCTION__, "NOT NATed - Endpoint independent");
      report += "NOT NATed - Endpoint independent.";
}
```

4.4 Esquema completo de la solución

Como conclusión y resumen, se muestra un diagrama completo con un cliente, un NAT y un servidor. Paralelizando varios de estos componentes y encadenando adecuadamente los filtros de captura de cada una de las partes, se pueden crear escenarios más complejos como se verá más adelante en este documento:

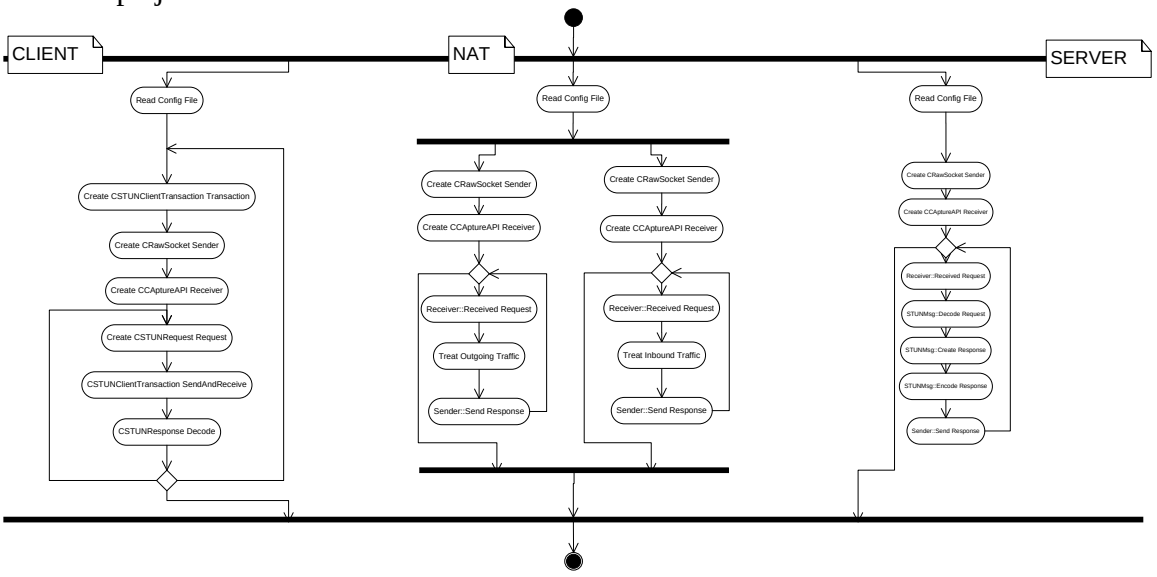

Figura 38. Esquema completo del diseño

El modo de funcionamiento en paralelo, creando componentes independientes mediante hilos de ejecución diferentes, permite aislar cada componente del resto y por lo tanto conseguir que aun cuando todos estén en ejecución en el mismo procesador, actúen como si se tratara de componentes remotos.

Uniendo esta característica de los hilos a la posibilidad de capturar tráfico de red a nivel UDP y enviar información incluyendo las cabeceras IP y UDP en los paquetes de salida, conseguimos una simulación casi real.

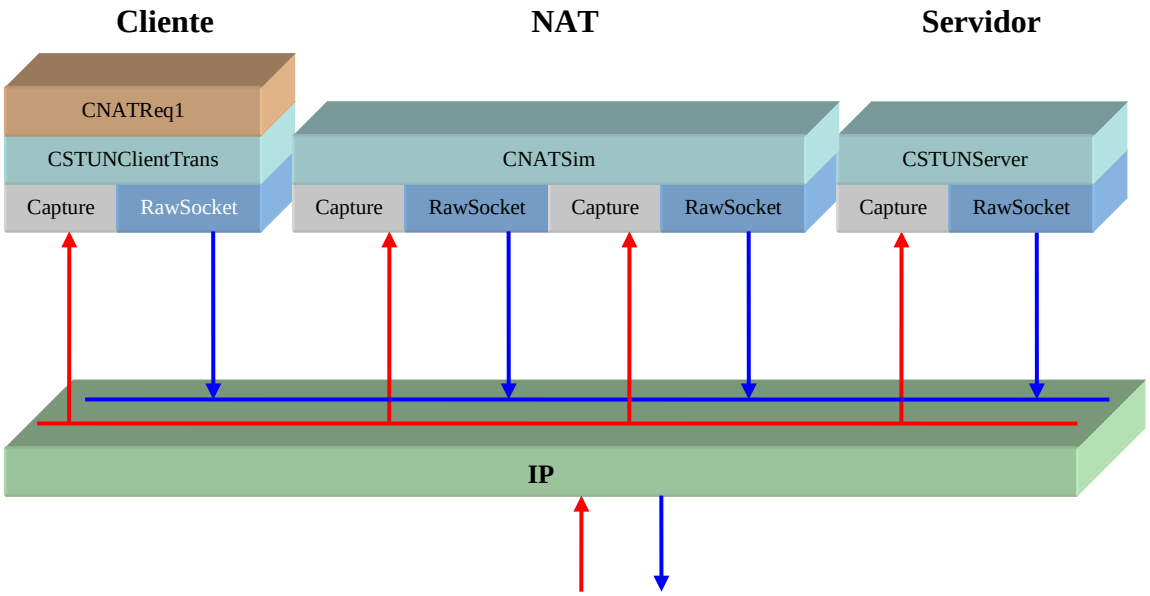

Figura 39. Esquema gráfico del diseño

Capítulo 5

Validación

5.1 Introducción

Para validar la solución software propuesta en los capítulos anteriores crearemos una serie de pruebas que permitan comprobar algunos de los requisitos que debe cumplir el NAT de acuerdo a las recomendaciones del grupo behave del área de transporte del IETF.

El capítulo está dividido en dos partes, la primera dedicada a la descripción de las pruebas desde el punto de vista del mecanismo NAT y la segunda parte muestra la descripción de la aplicación desarrollada para la validación de estas pruebas.

5.2 Pruebas de conformidad del mecanismo NAT

Para la validación del mecanismo NAT se han seleccionado los requisitos 1, 2, 3, 5, 6 y 8 descritos en el apartado *2.15 Requisitos NAT* de este documento, para comprobar que la herramienta de simulación se comporta de la misma manera que un escenario real de

conexión a través de NAT. En los siguientes apartados se recuerdan brevemente estos requerimientos y se describe el flujo de mensajes necesarios para determinar si el NAT cumple con las recomendaciones del IETF.

Aunque los diagramas de flujo son de muy fácil interpretación, se comentará una parte que aparece en todos ellos para facilitar su comprensión. En la siguiente figura se muestra el envío de un mensaje STUN desde el cliente hacia el servidor de tipo BINDING REQUEST:

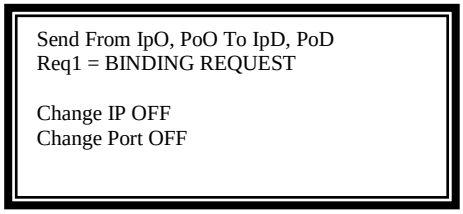

Figura 40. Envío de mensaje STUN.

En este mensaje se observa que la dirección origen es IpO y puerto origen es PoO, la dirección y puerto de destino son IpD y PoD respectivamente. El tipo de mensaje es BINDING REQUEST y no están activadas las banderas de cambio de IP ni de puerto.

El servidor al recibir este mensaje creará una respuesta STUN que será recibida por el cliente como la mostrada en la siguiente figura:

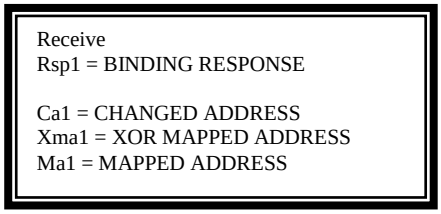

Figura 41. Envío de mensaje STUN.

En esta respuesta se observa que el tipo de mensaje recibido es BINDING RESPONSE y los atributos CHANGED ADDRESS, XOR MAPPED ADDRESS y MAPPED ADDRESS son Ca1, Xma1 y Ma1 respectivamente.

Para resaltar algunos cambios entre dos mensajes enviados por el cliente hacia el servidor, se utiliza el resaltado en negrita de los parámetros del mensaje que han sufrido algún cambio respecto del mensaje anterior.

5.2.1 Validación del tipo de mapeo

En esta validación vamos a determinar el tipo de mapeo del NAT. Recordemos que el NAT asigna las direcciones IP de que dispone en función de los cuatro parámetros que definen el origen y el destino de la comunicación, es decir, la dirección IP y el puerto origen y la dirección IP y el puerto de destino. En función de estos parámetros y de cómo se asignan las direcciones IP y el puerto públicos del NAT, diferenciamos tres tipos de NAT:

Independiente del extremo.
Dependiente de la dirección.
Dependiente de la dirección y del puerto.

Recordemos también lo que el IETF recomienda en este sentido. El primer requisito del grupo behave se refiere al tipo de mapeo (Ver apartado 2.15 de este documento):

*"Los NAT **deben** mapear las asociaciones como **independiente del extremo**."*

En el siguiente diagrama se muestra la secuencia de mensajes STUN necesarios para determinar el tipo de mapeo del NAT:

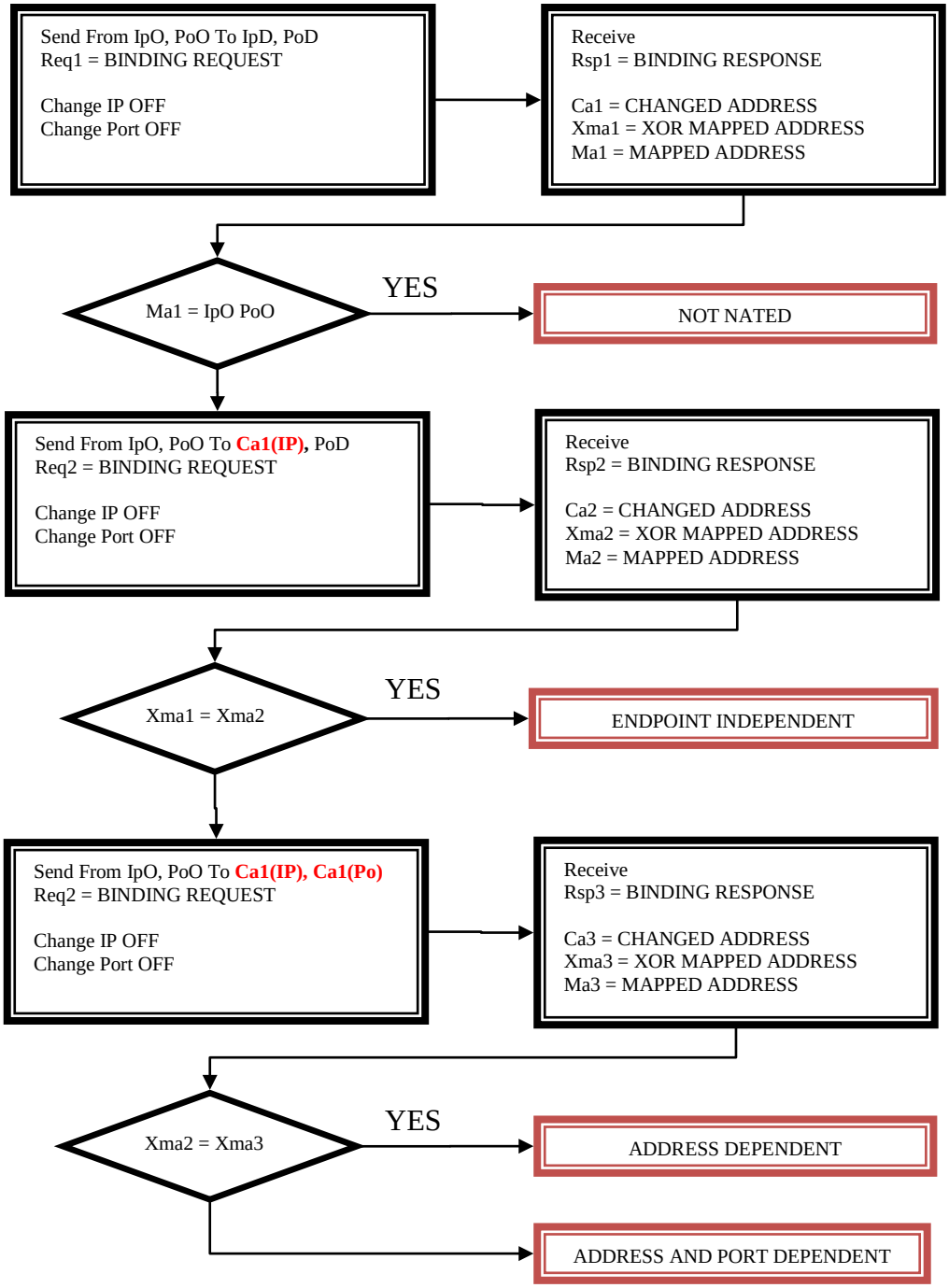

Figura 42. Validación del tipo de mapeo

5.2.2 Validación del tipo de filtro

En esta validación determinaremos el tipo de filtro de paquetes de entrada del NAT.

Como se comentó en el capítulo 2, apartado 2.8 Filtrado de paquetes de entrada, los posibles modos de funcionamiento del NAT en cuanto al filtro de paquetes de entrada son:

Independiente del extremo.
Dependiente de la dirección.
Dependiente de la dirección y del puerto.

Recordemos también lo que el IETF recomienda en este sentido. El octavo requisito del grupo behave se refiere al tipo de filtrado (Ver apartado 2.15 de este documento):

*"Si se considera de mayor importancia la **transparencia** de las aplicaciones, es **recomendable** que el NAT realiza un filtro **independiente del extremo**. Si es más importante que el NAT sea más **riguroso** a la hora de filtrar los paquetes, es **recomendable** que el NAT realice un filtro **dependiente de la dirección**.*

La configuración del nivel de filtro puede ser configurable por el administrador del NAT."

Para realizar el test, el servidor STUN debe soportar el cambio de dirección y puerto en el envío de las respuestas. Esto es, el cliente solicitará al servidor que el origen de la respuesta sea diferente del destino de su petición. De esta manera el NAT asociará una dirección IP y un puerto para la comunicación entre los dos extremos y al recibir la respuesta del servidor, el origen de esta respuesta será distinto del destino del mensaje que transitó por el NAT hacia la zona pública. Si el mensaje de respuesta, cuyo origen es desconocido por el NAT puesto que la petición iba con otra dirección de destino, es recibido por el cliente en la zona privada, el NAT habrá permitido la entrada de paquetes desde direcciones IP a las que no ha enviado ningún mensaje.

En el siguiente diagrama se muestra la secuencia de mensajes STUN necesarios para determinar el tipo de filtro del NAT. En esta secuencia se ha denotado con NULL la ausencia de mensaje de respuesta. Además se han omitido los reintentos de la capa STUN:

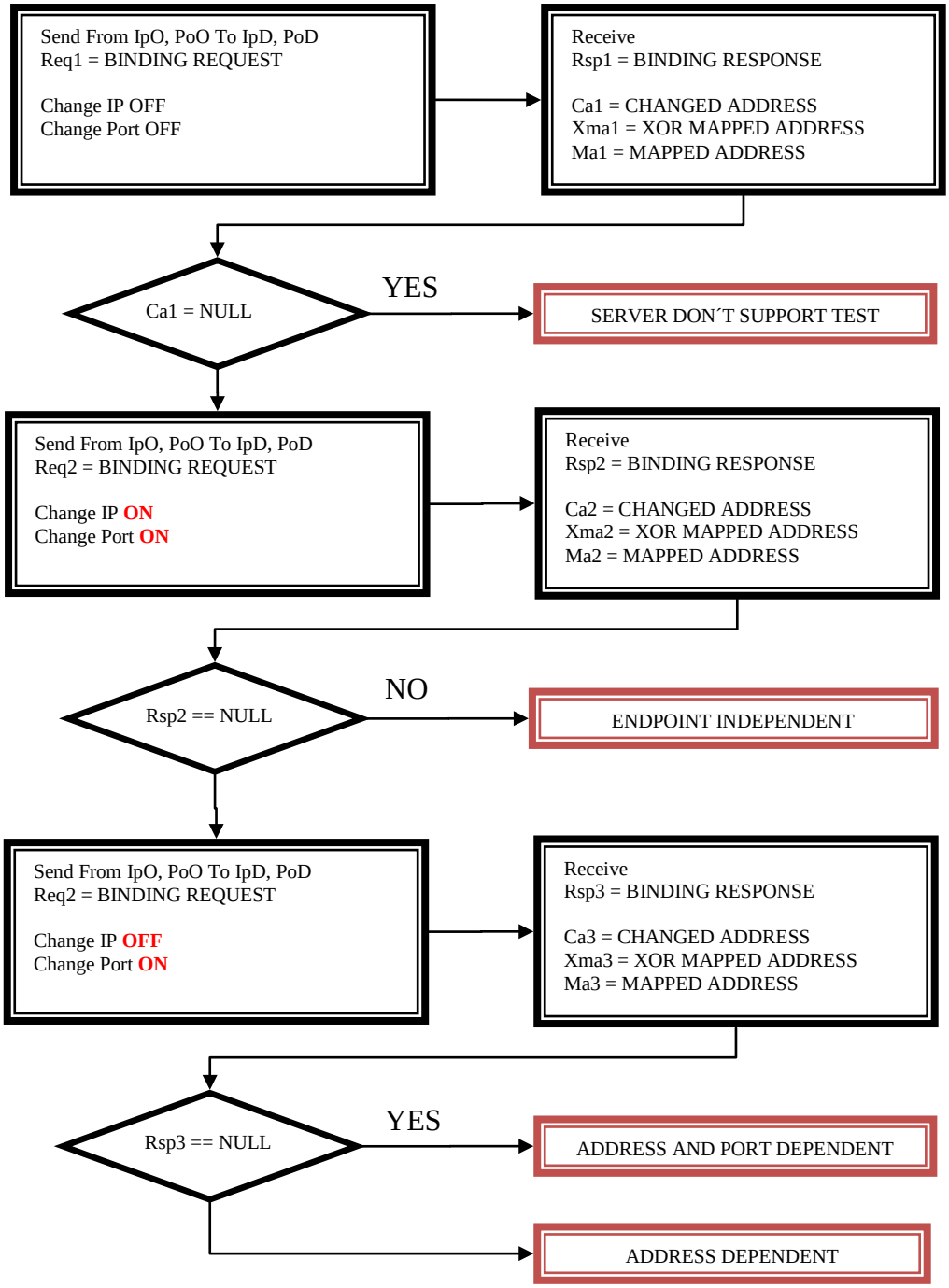

Figura 43. Validación del tipo de filtro

5.2.3 Validación del tiempo de refresco

En esta validación determinaremos si el NAT es capaz de mantener activo la conexión entre el extremo privado y el extremo público durante un tiempo T en ausencia de tráfico.

Como se comentó en el capítulo 2, apartado 2.6 Refresco de mapeo:

"Algunos NAT mantienen el mapeo activo cuando un paquete viaja de la zona interna hacia la zona externa del NAT. A este comportamiento se le llama ***refresco de salida****.*

Otros NAT mantienen el mapeo activo cuando un paquete es enviado desde el exterior hacia el interior del NAT. Este comportamiento es llamado ***refresco de entrada****.*

También hay NAT que mantienen el mapeo activo en ambos casos.

El refresco de salida es necesario para que el cliente pueda mantener el mapeo activo.

El refresco de entrada puede ser necesario en aplicaciones que no tienen tráfico hacia el exterior".

Recordemos también lo que el IETF recomienda en este sentido. Los requisitos 5 y 6 del grupo behave se refieren al refresco de mapeo (Ver apartado 2.15 de este documento):

"Un mapeo UDP ***no debe*** *expirar en menos de dos minutos, a menos que se deba aplicar la excepción a este requisito.*

- *Para puertos destino bien conocidos* ***puede*** *expirar en un tiempo menor que el especificado por la aplicación que corre sobre ese puerto de destino.*
- *El valor de este tiempo en el NAT* ***puede*** *ser configurable.*
- *El valor recomendable por defecto para un mapeo UDP en el NAT es de cinco minutos".*

"El refresco de un mapeo en el NAT ***debe ser*** *de* ***salida****.*

Adicionalmente el refresco ***puede ser*** *de entrada".*

Para descubrir el tiempo de refresco, implementaremos el test propuesto por el grupo behave en el borrador **NAT Behavior Discovery Using STUN (**draft-ietf-behave-nat-behavior-discovery-05), en la sección 4.5. Binding Lifetime Discovery. En este test, el cliente envía un mensaje de descubrimiento hacia el servidor provocando la creación en el NAT de una entrada en la tabla de mapeos:

IP 1-Puerto 2 (Privado), IP 3-Puerto 4 (Público) ←→ IP 5-Puerto 6 (NAT)

A continuación extrae de la respuesta recibida la dirección asignada por el NAT a esta comunicación y espera un tiempo T para volver a enviar un mensaje hacia el mismo destino, cambiando el puerto origen, y conteniendo el atributo RESPONSE ADDRESS con el valor de la dirección extraída de la respuesta. Este atributo solicita al servidor que la respuesta sea enviada a la dirección contenida en el cuerpo del mensaje STUN en lugar de la dirección y puerto contenidos en las cabeceras IP y UDP respectivamente. Si la respuesta es recibida en el puerto utilizado para el envío primer mensaje, entonces el tiempo de permanencia de la conexión es superior a T, en otro caso este tiempo habrá expirado.

En el siguiente diagrama se muestra la secuencia de mensajes STUN necesarios para determinar el tiempo de mapeo del NAT:

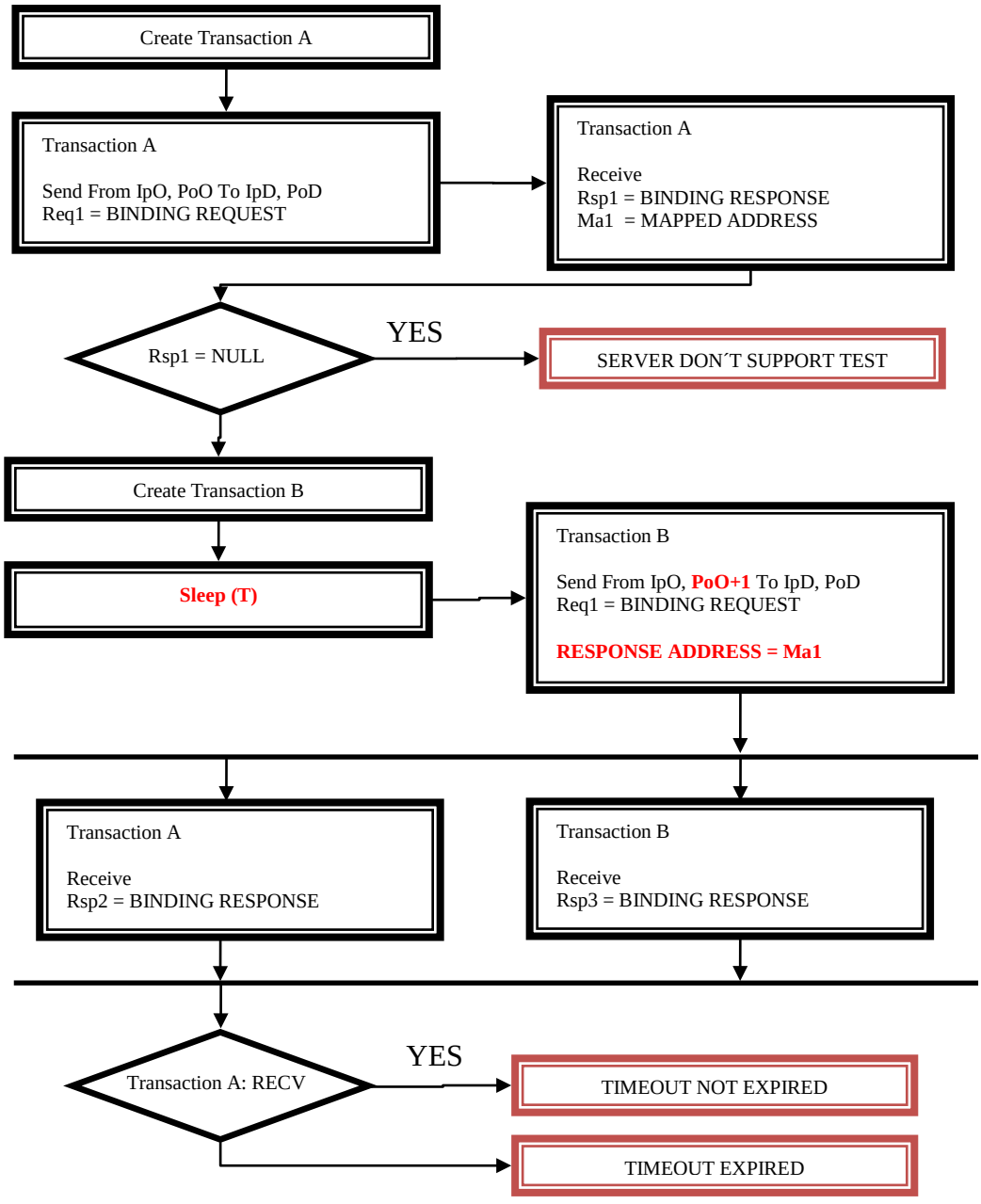

Figura 44. Validación del tiempo de refresco

5.2.4 Validación del tipo de asignación de IP

Recordemos que para aquellos NAT que disponen de varias direcciones IP públicas, la asignación de IP puede ser pareada o arbitraria.

La asignación pareada es aquella que asigna siempre la misma dirección IP pública para todas las conexiones procedentes de la misma dirección IP privada. La asignación arbitraria de IP pública es aquella que no tiene en cuenta la dirección IP privada y podía asignar cualquiera de las IP públicas de que dispone.

El IETF recomienda la asignación de IP pareada en su requisito número 2:

*"Es **recomendable** que los NAT tengan un modo **pareado** si disponen de un parque de direcciones IP externas."*

Para comprobar el comportamiento del NAT en este sentido se procederá a elaborar un test que determine esta característica mediante el envío reiterado de mensajes de descubrimiento. El número de mensajes mínimo que se considera suficiente para deducir que el test es fiable es a priori desconocido. Por esta razón, este número de mensajes se dejará a la elección del usuario. Si todas las respuestas a los mensajes de descubrimiento contienen la misma dirección IP pública asignada por el NAT, se dará por supuesto que el NAT funciona en modo pareado, siendo arbitrario en caso contrario.

En el siguiente diagrama se muestra la secuencia de mensajes STUN necesarios para determinar el tipo de mapeo del NAT:

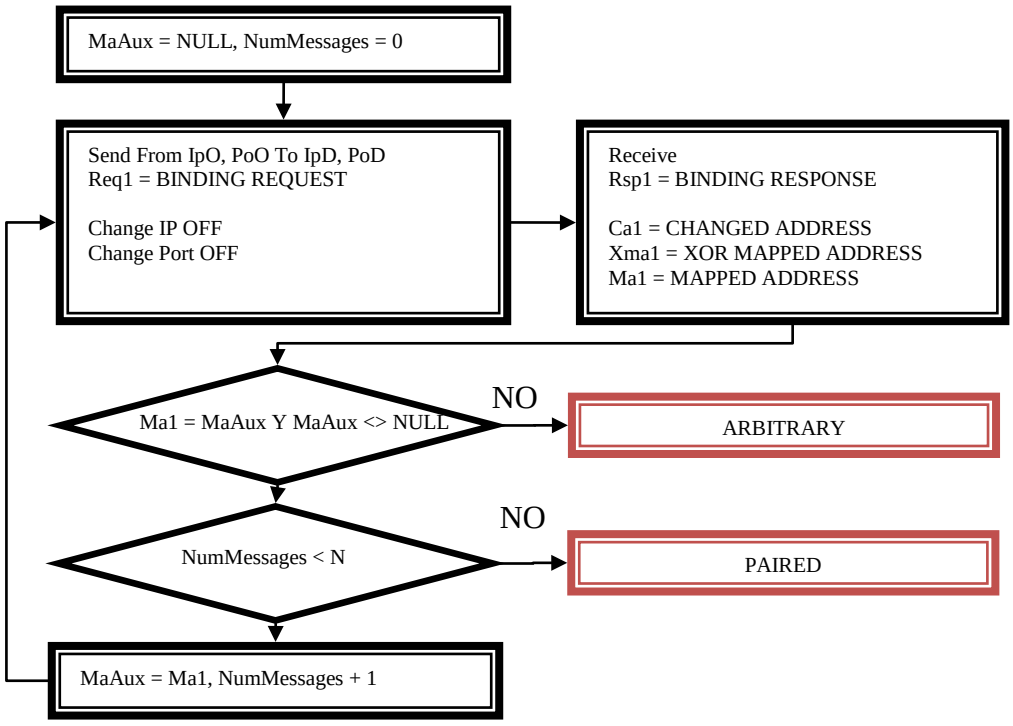

Figura 45. Validación del tipo de asignación de IP

5.2.5 Validación del tipo de asignación de puerto

Como se comentó en el apartado 2.5 de este documento, *"algunos NAT intentan preservar el número de puerto usado internamente en la red privada"*. Este tipo de asignación, denominada asignación conservativa, puede derivar en la sobrecarga de asignación de puertos. La sobrecarga de puertos implica que el NAT conserve el número de puerto incluso en ausencia de direcciones IP públicas con el puerto solicitado disponible. En este caso, el NAT da por finalizada la primera conexión y crea una nueva asociación para la nueva comunicación solicitada.

El IETF propone la eliminación de NAT de la sobrecarga de puertos en el requisito número 3:

*"El NAT **no debe** usar la **sobrecarga de puertos**.*

- *Si el puerto origen está en el rango (0-1023) es **recomendable** que el puerto origen del NAT esté en el mismo rango.*
- *Si el puerto origen esta en el rango (1024-65535) es **recomendable** que el puerto origen del NAT esté en el mismo rango."*

Para la validación de este requisito necesitaremos al menos dos clientes STUN. Ambos clientes enviarán mensajes de descubrimiento desde el mismo puerto origen. El cliente A al enviar el primer mensaje provocará la creación de una entrada en la tabla de asociación del NAT para el puerto solicitado solo si el NAT implementa la conservación de puertos. Si el cliente recibe en la respuesta del servidor el mismo puerto origen que utilizó en el envío de la petición podemos suponer que el NAT implementa la conservación del puerto. Si no es así, el test se da por finalizado y el NAT cumple con el requisito.

Para comprobar la sobrecarga de puertos deberemos enviar un nuevo mensaje desde un segundo cliente B, dentro del tiempo que el NAT mantiene activa la primera conexión y desde el mismo puerto origen. Si el NAT implementa la sobrecarga, deberá eliminar de la tabla de asociación la primera comunicación y asignar una nueva entrada para el cliente B y el puerto origen solicitado. Se puede observar que además el NAT no debe tener más direcciones IP externas libres si cuenta con un parque de direcciones IP. Por lo tanto el número de clientes necesarios para comprobar este requisito debe ser al menos mayor que el número de direcciones IP externas de las que dispone el NAT.

En el siguiente diagrama se muestra la secuencia de mensajes STUN necesarios para determinar el tipo de asignación de puertos del NAT:

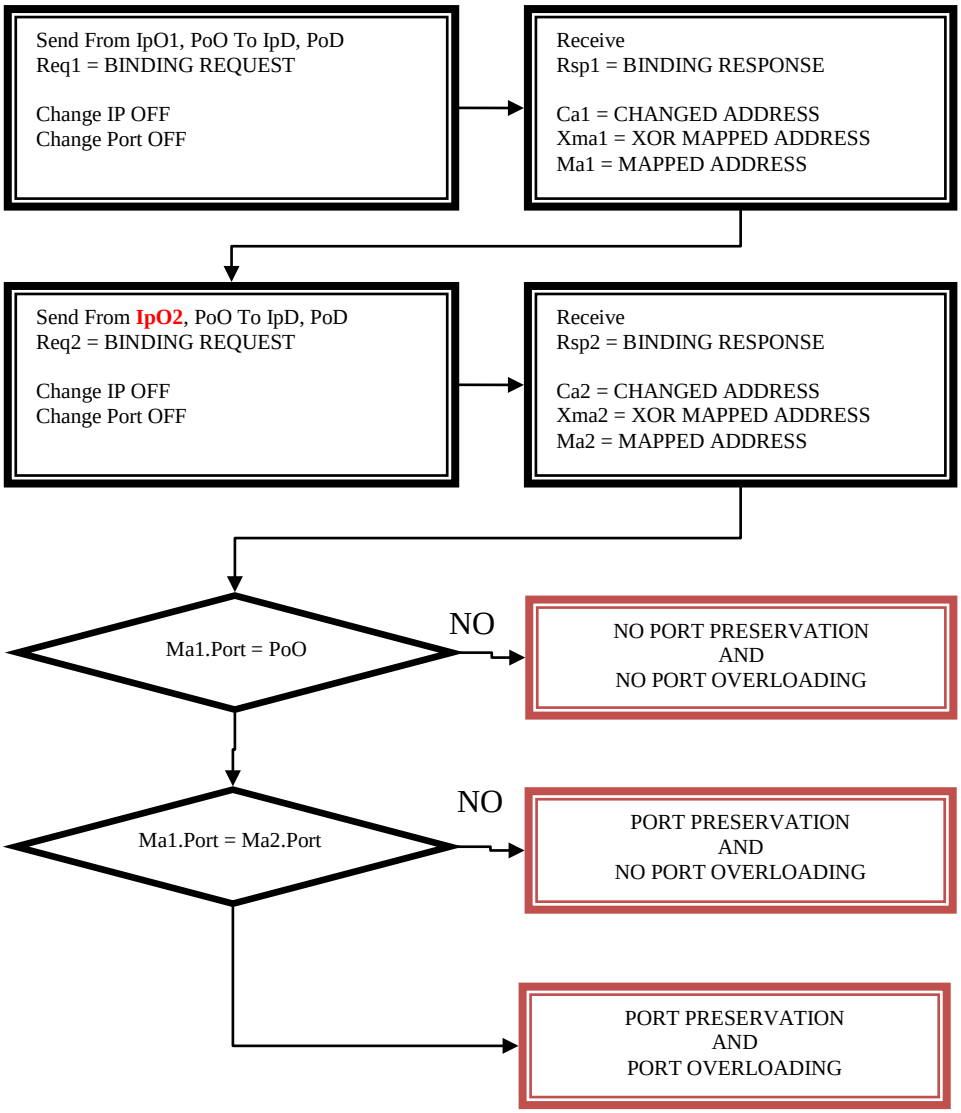

Figura 46. Validación del tipo de asignación de puerto

5.3 Pruebas globales del sistema

En este apartado describiremos la aplicación desarrollada para el estudio de los problemas derivados de la comunicación a través de NAT. Comenzaremos mostrando la instalación de la aplicación y a continuación se describirá su funcionamiento y configuración desde el punto de vista del usuario. También se describirán los ficheros de volcado de la información de depuración y finalmente se mostrará la ejecución de alguno de los test descritos en el capítulo anterior.

La aplicación NATSimulator es, como ya se comentó en los capítulos precedentes de análisis y diseño, una aplicación Windows basada en la librería MFC (Microsoft Foundation Class) y en el resto de librerías estáticas desarrolladas para el proyecto. Esta aplicación muestra un interfaz de múltiples documentos (MDI), basado en la arquitectura documento vista.

5.3.1 Instalación

Para la instalación de la aplicación NATSimulator necesitaremos haber instalado previamente la librería WinPCap 4.0.2. WinPCap es una librería de código abierto para la captura de paquetes y análisis de red en la plataforma Win32. Esta librería la encontramos en el CD de instalación de la aplicación y la podemos descargar desde la dirección http://www.winpcap.org.

Aunque no es necesario para la ejecución de la aplicación, es recomendable instalar también el analizador de protocolos Wireshark. Para el desarrollo del proyecto se ha utilizado la versión 0.99.8 (SVN Rev 24492) y se encuentra disponible en el CD de instalación. También se puede descargar una versión más reciente desde la dirección http://www.wireshark.org.

El proyecto se ha testeado en la plataforma Windows 2000 Service Pack 4, build 2195 y Windows 2003 Server Service Pack 2. Se ha utilizado esta plataforma debido a que el uso de sockets RAW está limitado en otras versiones del sistema operativo Windows. En la [MSDN] encontramos la siguiente nota:

"En Windows XP con Service Pack 2 (SP2) y Windows Vista, la posibilidad de enviar tráfico sobre sockets raw ha sido restringida de varias manera:

TCP no puede enviar datos sobre sockets raw.

Los datagramas UDP no se pueden enviar con una dirección de origen inválida. La IP origen para cualquier datagrama UDP de salida debe existir en un interfaz de red o el datagrama será descartado.

No está permitido llamar a la función bind con un socket raw.

Estas restricciones no son aplicables a Windows Server 2003 y Windows Server 2008 o para versiones anteriores a Windows XP con SP2."

Aun cuando existen estas limitaciones, es posible la ejecución de la aplicación sobre estas plataformas. Para ello es necesario instalar una maquina virtual y sobre ella un sistema operativo que permita el uso total de sockets raw, como son Windows 2000 o Windows 2003. En este escenario, la aplicación se ha testeado con éxito utilizando Windows Vista como sistema operativo principal y Windows 2000 corriendo sobre Virtual Box.

Para mantener la portabilidad del núcleo de la aplicación se ha utilizado la librería PThreads-Win32, que es una librería de código abierto que implementa el estándar [POSIX] 1003.1c 1995 para hilos en el entorno Microsoft Win32. Por lo tanto es

necesario contar con la librería pthreadVC2.dll en el mismo directorio en el que instalemos la aplicación. Esta librería se encuentra disponible en el CD de instalación de la aplicación y también podemos descargar una versión más actualizada desde la dirección http://sourceware.org/pthreads-win32.

Para finalizar, el procedimiento de instalación consta, sencillamente, de tres pasos:

1- Instalar WinPCap 4.0.2.
2- Instalar Wireshark 0.99.8 o posterior. Este paso es opcional.
3- Copiar los ficheros NATSimulator.exe y pthreadVC2.dll en el directorio deseado.

5.3.2 Creación de escenarios

Para la creación de escenarios de prueba se deberá crear un nuevo fichero desde el menú **File/New** o pulsando la secuencia de teclas Ctrl+N. Aparecerá una ventana de documento en blanco como la que se muestra a continuación:

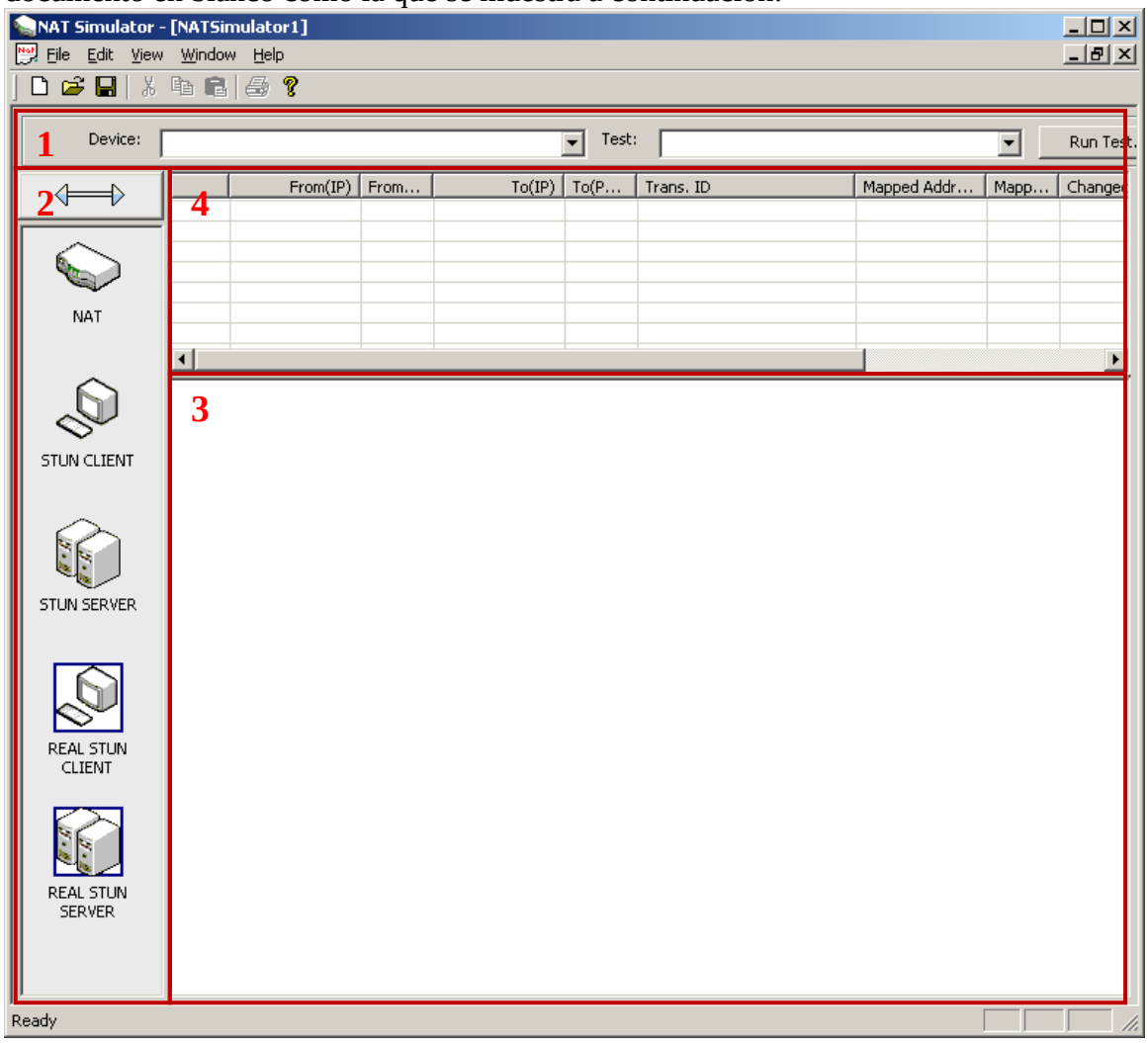

Figura 47. NAT Simulator. Creación de escenarios.

Esta ventana esta dividida en cuatro zonas bien diferenciadas:

Zona 1: en esta zona encontramos la selección de dispositivo de red y la selección y ejecución de los test de comunicación a través de NAT descritos en la sección anterior de este capítulo.

Zonas 2 y 3: área destinada a la creación del escenario. Pinchando sobre los elementos NAT, STUN CLIENT, STUN SERVER, REAL STUN CLIENT y REAL STUN SERVER podemos arrastrarlos a la zona 4, en la que se mostrará gráficamente la disposición de la red y se permitirá la configuración de los elementos que la componen. También disponemos del botón de creación de enlaces. La forma de enlazar dos elementos de simulación es arrastrándolos al escenario y pulsando sobre este botón de enlace, a continuación pinchamos sobre uno de los elementos de la red y sin soltar el botón del ratón lo llevaremos hacia el segundo elemento, donde liberaremos el ratón.

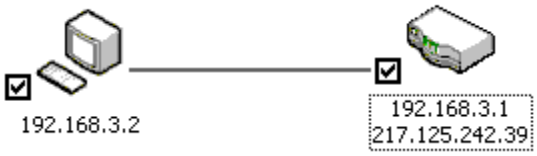

Figura 48. NAT Simulator. Enlace cliente-NAT

En la figura podemos observar dos elementos enlazados y sus direcciones IP. En la izquierda tenemos un cliente STUN con la dirección IP 192.168.3.2, conectado a la parte interna del NAT cuya dirección IP privada es 192.168.3.1 y una de las direcciones IP públicas de las que dispone es 217.125.242.39. Además se puede observar una caja de selección que sirve para realizar un seguimiento del tráfico de paquetes STUN que envía o recibe este elemento. Si esta caja esta seleccionada, todos los mensajes STUN aparecerán en la zona 3 a la hora de ejecutar el test.

En el enlace entre dos elementos de tipo NAT, se deberá indicar el sentido de la comunicación, diferenciando la zona privada de la zona pública de cada uno de los NAT. Esta característica del enlace entre dos NAT se resuelve mediante la aparición de un dialogo como el siguiente:

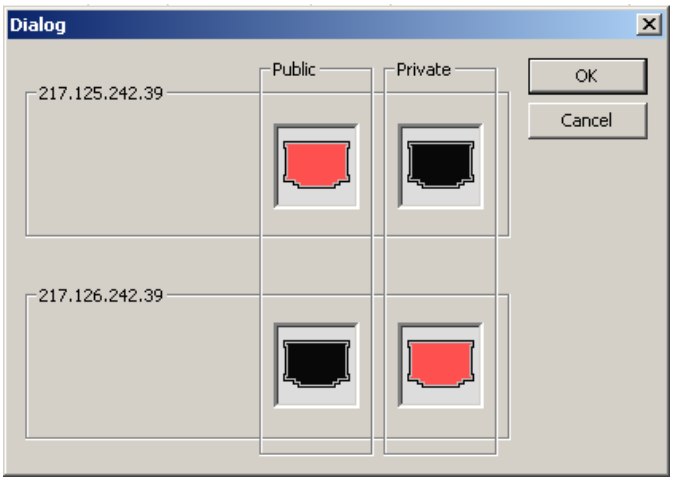

Figura 49. NAT Simulator. Dialogo enlace NAT-NAT.

Este dialogo muestra la dirección IP (la primera de que disponga en caso de que el NAT cuente con varias), de cada uno de los NAT enlazados. Representa un interfaz de

red público y uno privado para cada extremo del enlace. En el caso mostrado, el primer NAT con dirección IP pública 217.125.242.39 se enlaza con el NAT cuya dirección IP pública es 217.126.242.39 pero en su interfaz de red privada. De esta forma se pueden crear escenarios con NAT conectados en cascada. El escenario mostrará esta situación mediante una flecha de dirección en la línea de enlace como se observa a continuación:

Figura 50. NAT Simulator. Enlace NAT-NAT

Los elementos de tipo REAL no necesitan ser enlazados. Cuando arrastramos un cliente de este tipo sobre la zona de creación, se inserta un cliente STUN con la dirección IP local y el puerto 10000 por defecto. Al arrastrar un servidor STUN REAL, se crea un servidor con la dirección 69.0.208.27 para el servidor A y 69.0.209.22 para el servidor B y los puertos 3478 y 3479, que pertenecen a un servidor STUN público. Haciendo doble click sobre estos elementos reales podemos cambiar la configuración por defecto.

Para la configuración de los elementos del escenario, se han habilitado diálogos que aparecen haciendo doble clic sobre el elemento que deseamos configurar. Existen tres diálogos de configuración diferentes que se corresponden con cada uno de los tipos de elementos de que disponemos.

Para la configuración del cliente solo es necesario introducir la dirección IP y el número de puerto que usaremos en la comunicación.

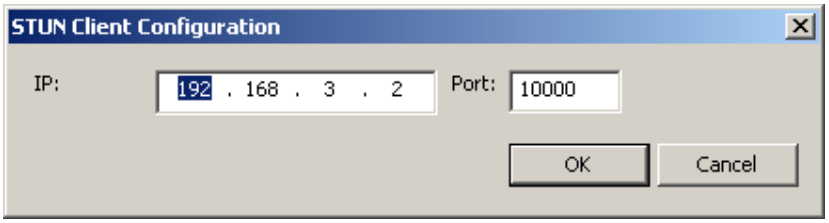

Figura 51. NAT Simulator. Configuración del cliente STUN.

Para la configuración del servidor STUN se dispone del correspondiente diálogo que permite introducir las direcciones IP y los puertos de escucha de peticiones STUN.

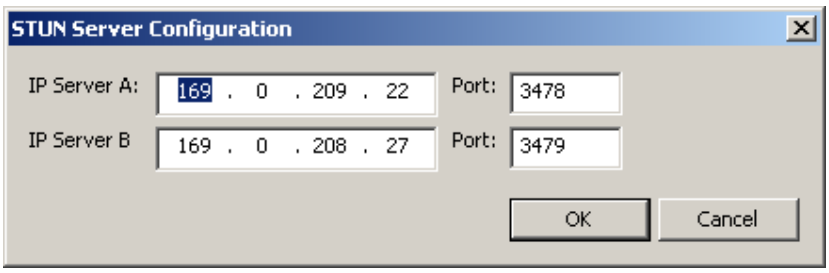

Figura 52. NAT Simulator. Configuración del servidor STUN.

Para la configuración del NAT contamos con el siguiente dialogo:

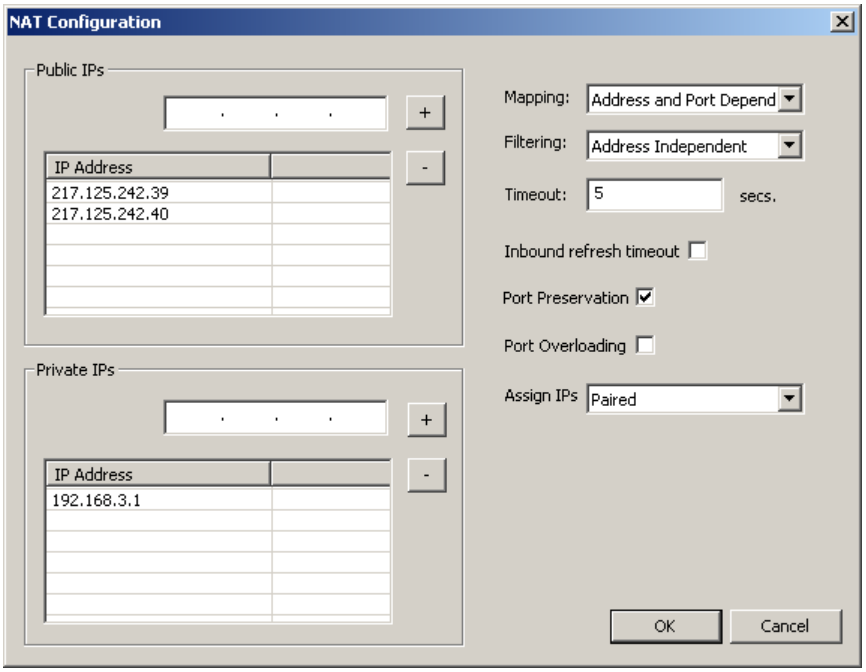

Figura 53. NAT Simulator. Configuración del NAT.

En este dialogo podemos introducir las direcciones IP públicas y privadas del NAT, el tipo de mapeo, el tipo de filtrado, el tiempo de permanencia de la asociación en ausencia de tráfico, el refresco de este tiempo cuando hay tráfico de entrada, la conservación y sobrecarga de puertos y la asignación de IP pareada o aleatoria.

Finalmente disponemos de un menú contextual que aparecerá pinchando sobre el botón derecho del ratón en la zona 3 desde el que podremos entre otras cosas, eliminar elementos del escenario.

Una vez creado y configurado el escenario, podremos guardar una copia de este desde la opción de menú **File/Save**, donde podremos introducir el nombre y seleccionar la ubicación del escenario.

Zona 4: En esta zona se mostrarán los mensajes STUN que se envían y reciben cada uno de los elementos del escenario durante la ejecución del test. Para que un elemento muestre su tráfico de paquetes STUN basta con pinchar sobre la caja de selección correspondiente al elemento. Por defecto esta caja de selección está marcada, de forma que a la hora de ejecutar el test mostrará los mensajes que envía o recibe este elemento.

La información mostrada en esta lista de mensajes es, de izquierda a derecha, la dirección IP y puerto de origen y destino de la cabecera IP y UDP respectivamente, el identificador del mensaje STUN, la dirección asignada por el NAT del atributo MAPPED ADDRESS del mensaje, el atributo CHANGED ADDRESS añadido por el servidor a la respuesta y el atributo XOR MAPPED ADDRESS de la respuesta.

Pinchando sobre esta zona con el botón derecho del ratón aparece un menú contextual que permite limpiar la lista de mensajes actual.

	From(IP)	From...	To(IP)	To(P...	Trans. ID	Mapped Addr...	Mapp...	Changed Add...	Chan...	XOR Mapped ...	XOR ...
SEND	192.168.3.2	10000	169.0.209.22	3478	D6 A8 0F 08 D6 A8 0F 08 ...						
RECV	192.168.3.2	10000	169.0.209.22	3478	D6 A8 0F 08 D6 A8 0F 08 ...						
SEND	217.125.242.39	10000	169.0.209.22	3478	D6 A8 0F 08 D6 A8 0F 08 ...						
RECV	217.125.242.39	10000	169.0.209.22	3478	D6 A8 0F 08 D6 A8 0F 08 ...						
SEND	169.0.209.22	3478	217.125.242.39	10000	D6 A8 0F 08 D6 A8 0F 08 ...	217.125.242.39	10000	169.0.208.27	3479	15.213.253.47	36806
RECV	169.0.209.22	3478	217.125.242.39	10000	D6 A8 0F 08 D6 A8 0F 08 ...	217.125.242.39	10000	169.0.208.27	3479	15.213.253.47	36806
SEND	169.0.209.22	3478	192.168.3.2	10000	D6 A8 0F 08 D6 A8 0F 08 ...	217.125.242.39	10000	169.0.208.27	3479	15.213.253.47	36806
RECV	169.0.209.22	3478	192.168.3.2	10000	D6 A8 0F 08 D6 A8 0F 08 ...	217.125.242.39	10000	169.0.208.27	3479	15.213.253.47	36806

Figura 54. NAT Simulator. Mensajes STUN.

5.3.3 Ejecución del test

Una vez creado el escenario, ejecutaremos el test que seleccionaremos en el desplegable que aparece en la parte superior de la ventana. Si la máquina en la que estamos realizando los test dispone de varios dispositivos de red, seleccionaremos uno de ellos en el desplegable **Device** y a continuación pulsaremos sobre el botón **Run test**. Aparece un dialogo en el que seleccionaremos el origen u orígenes y el destino o destinos de la comunicación. Dependiendo del test seleccionado, este dialogo solicita la introducción de información adicional necesaria para su ejecución.

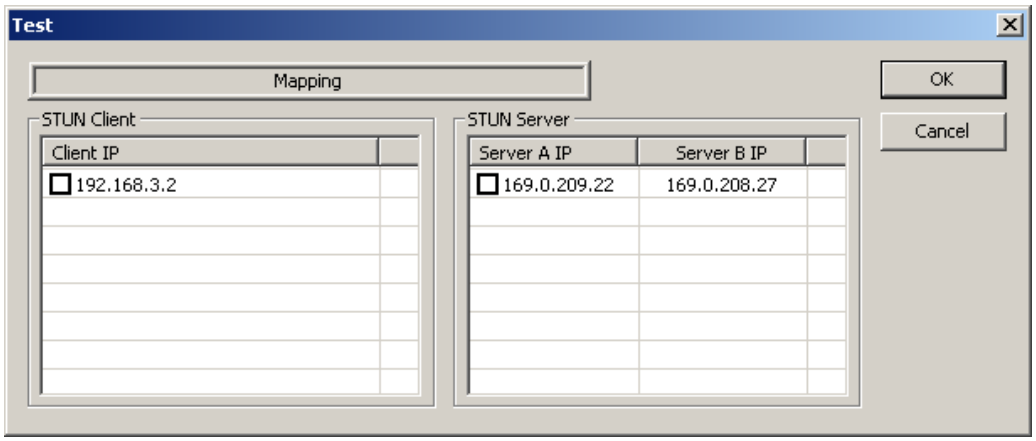

Figura 55. NAT Simulator. Dialogo de ejecución del test.

Durante la ejecución del test, aparece un indicador de actividad que desaparece cuando el test ha finalizado. Los elementos que han participado en el test muestran un icono distinto de aquellos que no han enviado ni recibido ningún mensaje como muestran las siguientes figuras:

Figura 56. NAT Simulator. Indicadores de test y tráfico.

5.3.4 Ficheros de configuración

Para la ejecución del escenario la aplicación genera una serie de ficheros de configuración que se guardan en el mismo directorio en el que tenemos instalado el ejecutable. Para cada NAT, se crea un fichero con nombre **NATSimN.config**, donde N es entero mayor que 0. Además, para cada servidor STUN se crea un fichero con nombre **STUNServerN.config** y por último se crea un único fichero donde se almacenan los clientes STUN con nombre **STUNClients.config**. Vamos a describir el contenido de estos ficheros.

Para la configuración de un NAT, son necesarios los siguientes parámetros:

```
#NATSim

MappingBehavior=AddressPortDependent
FilteringBehavior=AddressIndependent
TimeOut=5
IncommingRefreshTimeOut=0
PortPreservation=1
PortOverloading=0
IPAssignMode=Paired

PublicDev=\Device\NPF_{A255F475-686F-4421-AC57-0EA832445B02}
PrivateDev=\Device\NPF_{A255F475-686F-4421-AC57-0EA832445B02}

NATPrivateIP1=192.168.3.1
NATPublicIP1=217.125.242.39
NATPublicIP2=217.125.242.40

PrivateIP1=192.168.3.2
PublicIP1=169.0.209.22
PublicIP2=169.0.208.27
```

El primer bloque de parámetros indica el comportamiento del NAT en cuanto a la forma de asignar los mapeos, el filtro de paquetes de entrada, el tiempo de permanencia de la conexión en ausencia de tráfico, el indicador de refresco de mapeos con tráfico de entrada, la conservación y sobrecarga de puertos y la asignación pareada o aleatoria.

El segundo bloque contiene el dispositivo de red que utilizará en NAT para la escucha y captura de tráfico.

El tercer bloque muestra las direcciones IP privadas y públicas de las que dispone el NAT.

Por último, se enumeran las direcciones IP usadas en la zona privada del NAT y las direcciones IP alcanzables en la zona pública.

Para la configuración de un servidor STUN se crea un fichero como el mostrado a continuación:

```
#STUNServer

ServerDev=\Device\NPF_{A255F475-686F-4421-AC57-0EA832445B02}
ServerAIP=169.0.209.22
```

```
ServerBIP=169.0.208.27
ServerPort1=3478
ServerPort2=3479

#NATSim

NATPublicIP1=217.125.242.39
NATPublicIP2=217.125.242.40
```

En el primer bloque de parámetros se guardan el dispositivo de red usado por el servidor para la captura y envío de paquetes y las direcciones IP y puertos correspondientes a la pareja de servidores.

En el segundo bloque se encuentran las direcciones IP de los NAT que pueden alcanzar a este par de servidores.

Para los clientes STUN se crea un único fichero conteniendo la información del dispositivo de red usado por el cliente y su dirección IP y puerto de escucha del tráfico STUN:

```
#STUNClient
ClientADev=\Device\NPF_{A255F475-686F-4421-AC57-0EA832445B02}
ClientAIP=192.168.3.2
ClientAPort=10000
```

5.3.5 Ficheros de depuración

La ejecución de un test provoca el volcado a ficheros de depuración de la actividad de todos los elementos contenidos en el escenario. Estos ficheros tienen la extensión **log** y en este apartado vamos a dar las ideas básicas necesarias para interpretar el resultado de una ejecución de test. Para ello vamos a mostrar las trazas volcadas en estos ficheros de depuración para el test del tipo de mapeo de un escenario como el siguiente:

Figura 57. NAT Simulator. Escenario básico para depuración.

5.3.5.1 Fichero de depuración del servidor

Comenzaremos por el fichero de traza del servidor STUN, con nombre STUNServer1.log. Para facilitar la legibilidad del fichero se han omitido los campos iniciales en los que se vuelca la fecha y hora del la traza y el origen de la misma (nombre de clase y función dentro de la clase).

El inicio del fichero muestra la IP y la mascara de red:

```
Starting STUN server.
Net: 192.168.1.0.
Mask: 255.255.255.0.
```

A continuación aparece el filtro de paquetes que deberá capturar el servidor STUN:

```
Filter: udp and
       ( src host 217.125.242.39 or src host 217.125.242.40 )  and
       ( dst host 169.0.209.22 or dst host 169.0.208.27 ).
```

Después, comienza la captura de paquetes y se observa la llegada de un paquete STUN con origen el NAT y destino el primero de los servidores STUN. El paquete se traza primero en binario y para hacerlo más legible se ha formateado en tres partes correspondientes a las cabeceras IP, UDP y el paquete STUN:

```
Starting capture on device \Device\NPF_{A255F475-686F-4421-AC57-0EA832445B02}.
Capturing packes from 217.125.242.39 217.125.242.40  to 169.0.209.22 169.0.208.27 .
HANDLE 0x00093C

45 00 00 38 E1 10 00 00 64 11 2F E8 D9 7D F2 27 A9 00 D1 16
27 10 0D 96 00 24 00 00
00 01 00 08 9C 16 52 09 9C 16 52 09 1A 3A 00 00 1A 3A 00 00 00 03 00 04 00 00 00 00
```

Ahora vemos este mismo paquete interpretado por la capa STUN:

```
From 217.125.242.39:10000 To 169.0.209.22:3478
**************************
BINDING-REQUEST
--------------------------
CHANGE_REQUEST
Change IP: 0 Change Port: 0
**************************
```

Una vez procesada la petición, se traza el paquete STUN de respuesta, primero desde la capa STUN y después en crudo:

```
**************************
BINDING-RESPONSE
--------------------------
MAPPED_ADDRESS
217.125.242.39.10000
--------------------------
SOURCE_ADDRESS
169.0.209.22.3478
--------------------------
CHANGED_ADDRESS
169.0.208.27.3479
--------------------------
XOR_MAPPED_ADDRESS
69.107.160.46.12684
**************************

HANDLE 0x00093C From 169.0.209.22:3478 to 217.125.242.39:10000
45 00 60 00 E1 10 00 00 64 11 00 00 A9 00 D1 16 D9 7D F2 27
0D 96 27 10 00 4C 00 00
01 01 00 30 9C 16 52 09 9C 16 52 09 1A 3A 00 00
1A 3A 00 00 00 01 00 08 00 01 27 10 D9 7D F2 27
00 04 00 08 00 01 0D 96 A9 00 D1 16 00 05 00 08
00 01 0D 97 A9 00 D0 1B 80 20 00 08 00 01 31 8C
45 6B A0 2E
```

A partir de este punto se reciben dos nuevas peticiones y se envían sus respectivas respuestas. La primera petición procede de una dirección IP distinta de la anterior, además la IP de destino es la del servidor secundario y el puerto de destino es el mismo que en la petición anterior:

```
45 00 00 38 E1 10 00 00 64 11 30 E2 D9 7D F2 28 A9 00 D0 1B
27 10 0D 96 00 24 00 00
00 01 00 08 1B 24 52 09 1B 24 52 09 28 3A 00 00 28 3A 00 00 00 03 00 04 00 00 00 00

From 217.125.242.40:10000 To 169.0.208.27:3478
```

```
**************************
BINDING-REQUEST
--------------------------
CHANGE_REQUEST
Change IP: 0 Change Port: 0
**************************
**************************
BINDING-RESPONSE
--------------------------
MAPPED_ADDRESS
217.125.242.40.10000
--------------------------
SOURCE_ADDRESS
169.0.208.27.3478
--------------------------
CHANGED_ADDRESS
169.0.209.22.3479
--------------------------
XOR_MAPPED_ADDRESS
194.89.160.33.779
**************************

HANDLE 0x00093C From 169.0.208.27:3478 to 217.125.242.40:10000
45 00 60 00 E1 10 00 00 64 11 00 00 A9 00 D0 1B D9 7D F2 28 0D 96 27 10 00 4C 00 00 01 01
00 30 1B 24 52 09 1B 24 52 09 28 3A 00 00 28 3A 00 00 00 01 00 08 00 01 27 10 D9 7D F2 28
00 04 00 08 00 01 0D 96 A9 00 D0 1B 00 05 00 08 00 01 0D 97 A9 00 D1 16 80 20 00 08 00 01
03 0B C2 59 A0 21
```

La segunda petición procede de la misma IP que la primera pero diferente puerto. El destino del mensaje es la IP y puerto secundarios del servidor STUN:

```
45 00 00 38 E1 10 00 00 64 11 30 E3 D9 7D F2 27 A9 00 D0 1B 00 00 0D 97 00 24 00 00 00 01
00 08 48 25 52 09 48 25 52 09 28 3A 00 00 28 3A 00 00 00 03 00 04 00 00 00 00

From 217.125.242.39:0 To 169.0.208.27:3479
**************************
BINDING-REQUEST
--------------------------
CHANGE_REQUEST
Change IP: 0 Change Port: 0
**************************
**************************
BINDING-RESPONSE
--------------------------
MAPPED_ADDRESS
217.125.242.39.0
--------------------------
SOURCE_ADDRESS
169.0.208.27.3479
--------------------------
CHANGED_ADDRESS
169.0.209.22.3478
--------------------------
XOR_MAPPED_ADDRESS
145.88.160.46.9544
**************************

HANDLE 0x00093C From 169.0.208.27:3479 to 217.125.242.39:0
45 00 60 00 E1 10 00 00 64 11 00 00 A9 00 D0 1B D9 7D F2 27 0D 97 00 00 00 4C 00 00 01 01
00 30 48 25 52 09 48 25 52 09 28 3A 00 00 28 3A 00 00 00 01 00 08 00 01 00 00 D9 7D F2 27
00 04 00 08 00 01 0D 97 A9 00 D0 1B 00 05 00 08 00 01 0D 96 A9 00 D1 16 80 20 00 08 00 01
25 48 91 58 A0 2E

Stoping first STUN server.
HANDLE 0x00093C
Stoping capture from 217.125.242.39 217.125.242.40  to 169.0.209.22 169.0.208.27 .
```

5.3.5.2 Fichero de depuración del cliente

Ahora vamos a fijarnos en la traza volcada por el cliente STUN en el fichero Mapping.log. El nombre del fichero es dependiente del test realizado y obviamente en este caso se trata del tipo de mapeo realizado por el NAT.

Las primeras líneas del fichero muestran el test realizado y la red del cliente:

```
Starting mapping behavior test...
Net: 192.168.1.0.
Mask: 255.255.255.0.
```

A continuación se muestra el filtro para la captura de paquetes UDP, en este caso serán aquellos con dirección de destino 192.168.3.2 y el puerto es el 10000:

```
Filter: udp and ( dst host 192.168.3.2 )  and ( dst port 10000 ).
Starting capture on device \Device\NPF_{A255F475-686F-4421-AC57-0EA832445B02}. Capturing
packes from all to 192.168.3.2.
```

Ahora se envía una petición de descubrimiento hacia el servidor con dirección 169.0.209.22 y el puerto 3478:

```
HANDLE 0x0008EC
**************************
BINDING-REQUEST
--------------------------
CHANGE_REQUEST
Change IP: 0 Change Port: 0
**************************
HANDLE 0x0008EC From 192.168.3.2:10000 to 169.0.209.22:3478
45 00 38 00 E1 10 00 00 64 11 00 00 C0 A8 03 02 A9 00 D1 16 27 10 0D 96 00 24 00 00 00 01
00 08 3B 50 CA 02 3B 50 CA 02 F0 1A 00 00 F0 1A 00 00 00 03 00 04 00 00 00 00
```

El servidor responde a la petición indicando que la dirección mapeada por el NAT es 217.125.242.39 y el puerto 10000. Además indica que posee una dirección alternativa en 169.0.208.27 y puerto 3479. Incluye también el XOR de la dirección mapeada por el NAT:

```
45 00 00 60 E1 10 00 00 64 11 37 BB A9 00 D1 16 C0 A8 03 02 0D 96 27 10 00 4C 00 00 01 01
00 30 3B 50 CA 02 3B 50 CA 02 F0 1A 00 00 F0 1A 00 00 00 01 00 08 00 01 27 10 D9 7D F2 27
00 04 00 08 00 01 0D 96 A9 00 D1 16 00 05 00 08 00 01 0D 97 A9 00 D0 1B 80 20 00 08 00 01
77 2B E2 2D 38 25
From 169.0.209.22:3478 To 192.168.3.2:10000

**************************
BINDING-RESPONSE
--------------------------
MAPPED_ADDRESS
217.125.242.39.10000
--------------------------
SOURCE_ADDRESS
169.0.209.22.3478
--------------------------
CHANGED_ADDRESS
169.0.208.27.3479
--------------------------
XOR_MAPPED_ADDRESS
226.45.56.37.30507
**************************
```

El cliente envía una nueva petición al servidor alternativo, pero conservando el puerto del primer servidor, esto es, cambiando únicamente la dirección de destino:

```
**************************
BINDING-REQUEST
--------------------------
CHANGE_REQUEST
Change IP: 0 Change Port: 0
**************************
```

```
HANDLE 0x0008EC From 192.168.3.2:10000 to 169.0.208.27:3478
45 00 38 00 E1 10 00 00 64 11 00 00 C0 A8 03 02 A9 00 D0 1B 27 10 0D 96 00 24 00 00 00 01
00 08 90 51 CA 02 90 51 CA 02 F3 1A 00 00 F3 1A 00 00 00 03 00 04 00 00 00 00
```

El servidor alternativo responde la dirección mapeada por el NAT como 217.125.242.40 y puerto 10000. Esto quiere decir que el tipo de mapeo del NAT es cuanto menos dependiente de la dirección:

```
45 00 00 60 E1 10 00 00 64 11 38 B6 A9 00 D0 1B C0 A8 03 02 0D 96 27 10 00 4C 00 00 01 01
00 30 90 51 CA 02 90 51 CA 02 F3 1A 00 00 F3 1A 00 00 00 01 00 08 00 01 27 10 D9 7D F2 28
00 04 00 08 00 01 0D 96 A9 00 D0 1B 00 05 00 08 00 01 0D 97 A9 00 D1 16 80 20 00 08 00 01
76 80 49 2C 38 2A
From 169.0.208.27:3478 To 192.168.3.2:10000
**************************
BINDING-RESPONSE
--------------------------
MAPPED_ADDRESS
217.125.242.40.10000
--------------------------
SOURCE_ADDRESS
169.0.208.27.3478
--------------------------
CHANGED_ADDRESS
169.0.209.22.3479
--------------------------
XOR_MAPPED_ADDRESS
73.44.56.42.30336
**************************
```

Ya solo queda por determinar si además el NAT es dependiente del puerto. Para ello envía una última petición hacia el servidor y puerto alternativos.

```
**************************
BINDING-REQUEST
--------------------------
CHANGE_REQUEST
Change IP: 0 Change Port: 0
**************************
HANDLE 0x0008EC From 192.168.3.2:10000 to 169.0.208.27:3479
45 00 38 00 E1 10 00 00 64 11 00 00 C0 A8 03 02 A9 00 D0 1B 27 10 0D 97 00 24 00 00 00 01
00 08 C6 52 CA 02 C6 52 CA 02 F3 1A 00 00 F3 1A 00 00 00 03 00 04 00 00 00 00
```

El servidor responde con una dirección y puerto de mapeo diferente de las dos anteriores, en este caso es 217125.242.39 y puerto 0. Esto nos indica que el NAT es dependiente de la dirección y del puerto:

```
45 00 00 60 E1 10 00 00 64 11 38 B6 A9 00 D0 1B C0 A8 03 02 0D 97 27 10 00 4C 00 00 01 01
00 30 C6 52 CA 02 C6 52 CA 02 F3 1A 00 00 F3 1A 00 00 00 01 00 08 00 01 00 00 D9 7D F2 27
00 04 00 08 00 01 0D 97 A9 00 D0 1B 00 05 00 08 00 01 0D 96 A9 00 D1 16 80 20 00 08 00 01
52 C6 1F 2F 38 25
From 169.0.208.27:3479 To 192.168.3.2:10000
**************************
BINDING-RESPONSE
--------------------------
MAPPED_ADDRESS
217.125.242.39.0
--------------------------
SOURCE_ADDRESS
169.0.208.27.3479
--------------------------
CHANGED_ADDRESS
169.0.209.22.3478
--------------------------
XOR_MAPPED_ADDRESS
31.47.56.37.21190
**************************
```

Finalmente se muestra el resultado del test:

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>   Address and Port dependent mapping
```

```
HANDLE 0x0008EC
Stoping capture from all to 192.168.3.2 .
Terminating mapping behavior test...
```

5.3.5.3 Fichero de depuración del NAT

Para completar la explicación de los ficheros de depuración, explicaremos el volcado del NAT sobre el fichero NATSim1.log.

El fichero comienza volcando los parámetros de inicialización del NAT, leídos del fichero de configuración del NAT:

```
Mapping Behavior AddressPortDependant.
Filtering Behavior AddressIndependent.
Timeout 5000.
Incomming Refresh Timeout false.
Port Preservation true.
Port Overloading false.
IP Assign Mode Paired.
```

A continuación inicia los dos hilos de captura de tráfico público y de tráfico privado.

```
Starting capture of public traffic.
Starting capture of private traffic.
Net: 192.168.1.0.
Mask: 255.255.255.0.
Net: 192.168.1.0.
Mask: 255.255.255.0.
```

Ahora se observan los filtros de captura de la zona privada y de la zona pública:

```
Filter: udp and ( src host 169.0.209.22 or src host 169.0.208.27 )  and
                 ( dst host 217.125.242.39 or dst host 217.125.242.40 ).

Starting capture on device \Device\NPF_{A255F475-686F-4421-AC57-0EA832445B02}.
Capturing packes from 169.0.209.22 169.0.208.27  to 217.125.242.39 217.125.242.40 .

Filter: udp and ( src host 192.168.3.2 )  and
                ( dst host 169.0.209.22 or dst host 169.0.208.27 ) .
Starting capture on device \Device\NPF_{A255F475-686F-4421-AC57-0EA832445B02}.
Capturing packes from 192.168.3.2  to 169.0.209.22 169.0.208.27 .

HANDLE 0x00098C
HANDLE 0x00097C
```

Ahora aparece un paquete recibido desde la dirección 192.168.3.2 y puerto 10000 hacía la dirección 169.0.209.22 y puerto 3478:

```
45 00 00 38 E1 10 00 00 64 11 37 E3 C0 A8 03 02 A9 00 D1 16 27 10 0D 96 00 24 00 00 00 01
00 08 3B 50 CA 02 3B 50 CA 02 F0 1A 00 00 F0 1A 00 00 00 03 00 04 00 00 00 00
From 192.168.3.2:10000 To 169.0.209.22:3478
```

El NAT selecciona una dirección IP y puerto libre y actualiza las tablas internas de tráfico de salida y tráfico de entrada:

```
Inserted entry into outgoing table. From 192.168.3.2:10000 To 169.0.209.22:3478.
Inserted entry into incomming table. From 0.0.0.0:65536 To 217.125.242.39:10000.
```

Se envía y se traza el paquete modificado con la nueva dirección y puerto origen:

```
HANDLE 0x00097C From 217.125.242.39:10000 to 169.0.209.22:3478
45 00 38 00 E1 10 00 00 64 11 00 00 D9 7D F2 27 A9 00 D1 16 27 10 0D 96 00 24 00 00 00 01
00 08 3B 50 CA 02 3B 50 CA 02 F0 1A 00 00 F0 1A 00 00 00 03 00 04 00 00 00 00
```

Después de reenviar el paquete se muestra el estado de las tablas internas del NAT. En este punto es necesario comentar que se ha tomado como dirección nula la 0.0.0.0 y representa cualquier dirección y el puerto nulo como 65536 y representa cualquier puerto:

```
Outgoing Table

192.168.3.2:10000 - 169.0.209.22:03478 : 217.125.242.39:10000

Incomming Table

0.0.0.0:65536  - 217.125.242.39:10000 : 192.168.3.2:10000
```

El NAT recibe un paquete desde la dirección 169.0.209.22 y el puerto 3478. El destino de este paquete es 217.125.242.39 en el puerto 10000, por lo tanto se lo entregará a la dirección privada 192.168.3.2 en el puerto 10000:

```
45 00 00 60 E1 10 00 00 64 11 2F C0 A9 00 D1 16 D9 7D F2 27 0D 96 27 10 00 4C 00 00 01 01
00 30 3B 50 CA 02 3B 50 CA 02 F0 1A 00 00 F0 1A 00 00 00 01 00 08 00 01 27 10 D9 7D F2 27
00 04 00 08 00 01 0D 96 A9 00 D1 16 00 05 00 08 00 01 0D 97 A9 00 D0 1B 80 20 00 08 00 01
77 2B E2 2D 38 25
From 169.0.209.22:3478 To 217.125.242.39:10000
```

Por lo tanto, el NAT modifica el paquete y lo reenvía modificado:

```
HANDLE 0x00098C From 169.0.209.22:3478 to 192.168.3.2:10000
45 00 60 00 E1 10 00 00 64 11 00 00 A9 00 D1 16 C0 A8 03 02 0D 96 27 10 00 4C 00 00 01 01
00 30 3B 50 CA 02 3B 50 CA 02 F0 1A 00 00 F0 1A 00 00 00 01 00 08 00 01 27 10 D9 7D F2 27
00 04 00 08 00 01 0D 96 A9 00 D1 16 00 05 00 08 00 01 0D 97 A9 00 D0 1B 80 20 00 08 00 01
77 2B E2 2D 38 25

Outgoing Table

192.168.3.2:10000 -    169.0.209.22:03478 :  217.125.242.39:10000

Incomming Table

0.0.0.0:65536 -  217.125.242.39:10000 :     192.168.3.2:10000
```

El NAT recibe un nuevo paquete de la misma dirección privada que el paquete anterior, pero con otra IP de destino. Al ser dependiente de la dirección y del puerto, crea un nuevo mapeo para esta conexión como se observa en el volcado de loas tablas internas:

```
45 00 00 38 E1 10 00 00 64 11 38 DE C0 A8 03 02 A9 00 D0 1B 27 10 0D 96 00 24 00 00 00 01
00 08 90 51 CA 02 90 51 CA 02 F3 1A 00 00 F3 1A 00 00 00 03 00 04 00 00 00 00
From 192.168.3.2:10000 To 169.0.208.27:3478

Inserted entry into outgoing table. From 192.168.3.2:10000 To 169.0.208.27:3478.
Inserted entry into incomming table. From 0.0.0.0:65536 To 217.125.242.40:10000.

HANDLE 0x00097C From 217.125.242.40:10000 to 169.0.208.27:3478
45 00 38 00 E1 10 00 00 64 11 00 00 D9 7D F2 28 A9 00 D0 1B 27 10 0D 96 00 24 00 00 00 01
00 08 90 51 CA 02 90 51 CA 02 F3 1A 00 00 F3 1A 00 00 00 03 00 04 00 00 00 00

Outgoing Table

192.168.3.2:10000 -    169.0.208.27:03478 :  217.125.242.40:10000
192.168.3.2:10000 -    169.0.209.22:03478 :  217.125.242.39:10000

Incomming Table

0.0.0.0:65536 -  217.125.242.39:10000 :     192.168.3.2:10000
0.0.0.0:65536 -  217.125.242.40:10000 :     192.168.3.2:10000
```

El resto de la traza es fácilmente interpretable con las explicaciones dadas hasta este punto:

```
45 00 00 60 E1 10 00 00 64 11 30 BA A9 00 D0 1B D9 7D F2 28 0D 96 27 10 00 4C 00 00 01 01
00 30 90 51 CA 02 90 51 CA 02 F3 1A 00 00 F3 1A 00 00 00 01 00 08 00 01 27 10 D9 7D F2 28
00 04 00 08 00 01 0D 96 A9 00 D0 1B 00 05 00 08 00 01 0D 97 A9 00 D1 16 80 20 00 08 00 01
76 80 49 2C 38 2A
From 169.0.208.27:3478 To 217.125.242.40:10000

HANDLE 0x00098C From 169.0.208.27:3478 to 192.168.3.2:10000
45 00 60 00 E1 10 00 00 64 11 00 00 A9 00 D0 1B C0 A8 03 02 0D 96 27 10 00 4C 00 00 01 01
00 30 90 51 CA 02 90 51 CA 02 F3 1A 00 00 F3 1A 00 00 00 01 00 08 00 01 27 10 D9 7D F2 28
00 04 00 08 00 01 0D 96 A9 00 D0 1B 00 05 00 08 00 01 0D 97 A9 00 D1 16 80 20 00 08 00 01
76 80 49 2C 38 2A

Outgoing Table

192.168.3.2:10000 -    169.0.208.27:03478 :  217.125.242.40:10000
192.168.3.2:10000 -    169.0.209.22:03478 :  217.125.242.39:10000

Incomming Table

0.0.0.0:65536 -  217.125.242.39:10000 :     192.168.3.2:10000
0.0.0.0:65536 -  217.125.242.40:10000 :     192.168.3.2:10000


45 00 00 38 E1 10 00 00 64 11 38 DE C0 A8 03 02 A9 00 D0 1B 27 10 0D 97 00 24 00 00 00 01
00 08 C6 52 CA 02 C6 52 CA 02 F3 1A 00 00 F3 1A 00 00 00 03 00 04 00 00 00 00
From 192.168.3.2:10000 To 169.0.208.27:3479

Inserted entry into outgoing table. From 192.168.3.2:10000 To 169.0.208.27:3479.
Inserted entry into incomming table. From 0.0.0.0:65536 To 217.125.242.39:0.

HANDLE 0x00097C From 217.125.242.39:0 to 169.0.208.27:3479
45 00 38 00 E1 10 00 00 64 11 00 00 D9 7D F2 27 A9 00 D0 1B 00 00 0D 97 00 24 00 00 00 01
00 08 C6 52 CA 02 C6 52 CA 02 F3 1A 00 00 F3 1A 00 00 00 03 00 04 00 00 00 00

Outgoing Table

192.168.3.2:10000 -    169.0.208.27:03478 :  217.125.242.40:10000
192.168.3.2:10000 -    169.0.208.27:03479 :  217.125.242.39:00000
192.168.3.2:10000 -    169.0.209.22:03478 :  217.125.242.39:10000

Incomming Table

0.0.0.0:65536 -  217.125.242.39:00000 :     192.168.3.2:10000
0.0.0.0:65536 -  217.125.242.39:10000 :     192.168.3.2:10000
0.0.0.0:65536 -  217.125.242.40:10000 :     192.168.3.2:10000


45 00 00 60 E1 10 00 00 64 11 30 BB A9 00 D0 1B D9 7D F2 27 0D 97 00 00 00 4C 00 00 01 01
00 30 C6 52 CA 02 C6 52 CA 02 F3 1A 00 00 F3 1A 00 00 00 01 00 08 00 01 00 00 D9 7D F2 27
00 04 00 08 00 01 0D 97 A9 00 D0 1B 00 05 00 08 00 01 0D 96 A9 00 D1 16 80 20 00 08 00 01
52 C6 1F 2F 38 25
From 169.0.208.27:3479 To 217.125.242.39:0

HANDLE 0x00098C From 169.0.208.27:3479 to 192.168.3.2:10000
45 00 60 00 E1 10 00 00 64 11 00 00 A9 00 D0 1B C0 A8 03 02 0D 97 27 10 00 4C 00 00 01 01
00 30 C6 52 CA 02 C6 52 CA 02 F3 1A 00 00 F3 1A 00 00 00 01 00 08 00 01 00 00 D9 7D F2 27
00 04 00 08 00 01 0D 97 A9 00 D0 1B 00 05 00 08 00 01 0D 96 A9 00 D1 16 80 20 00 08 00 01
52 C6 1F 2F 38 25

Outgoing Table

192.168.3.2:10000 -    169.0.208.27:03478 :  217.125.242.40:10000
192.168.3.2:10000 -    169.0.208.27:03479 :  217.125.242.39:00000
192.168.3.2:10000 -    169.0.209.22:03478 :  217.125.242.39:10000

Incomming Table

0.0.0.0:65536 -  217.125.242.39:00000 :     192.168.3.2:10000
0.0.0.0:65536 -  217.125.242.39:10000 :     192.168.3.2:10000
0.0.0.0:65536 -  217.125.242.40:10000 :     192.168.3.2:10000

Stoping capture of public traffic.
HANDLE 0x00097C
Stoping capture from 192.168.3.2  to 169.0.209.22 169.0.208.27 .
HANDLE 0x00098C
Stoping capture from 169.0.209.22 169.0.208.27  to 217.125.242.39 217.125.242.40 .
Stoping capture of private traffic.
```

5.3.6 Ejemplo de simulación. Tipo de mapeo

En este apartado vamos a mostrar algunas capturas de pantalla de la simulación básica de descubrimiento del tipo de mapeo variando la configuración del NAT para que se pueda observar como varía el intercambio de mensajes en función del tipo de NAT.

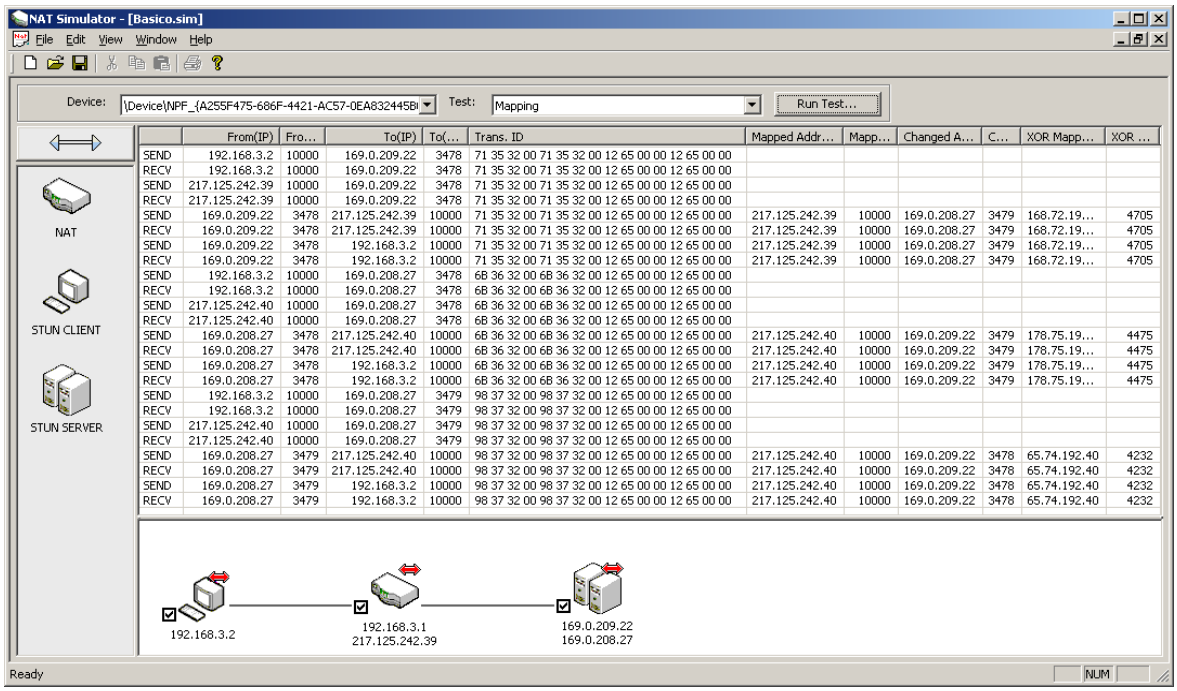

Figura 58. Validación del tipo de mapeo – Dependiente de la dirección

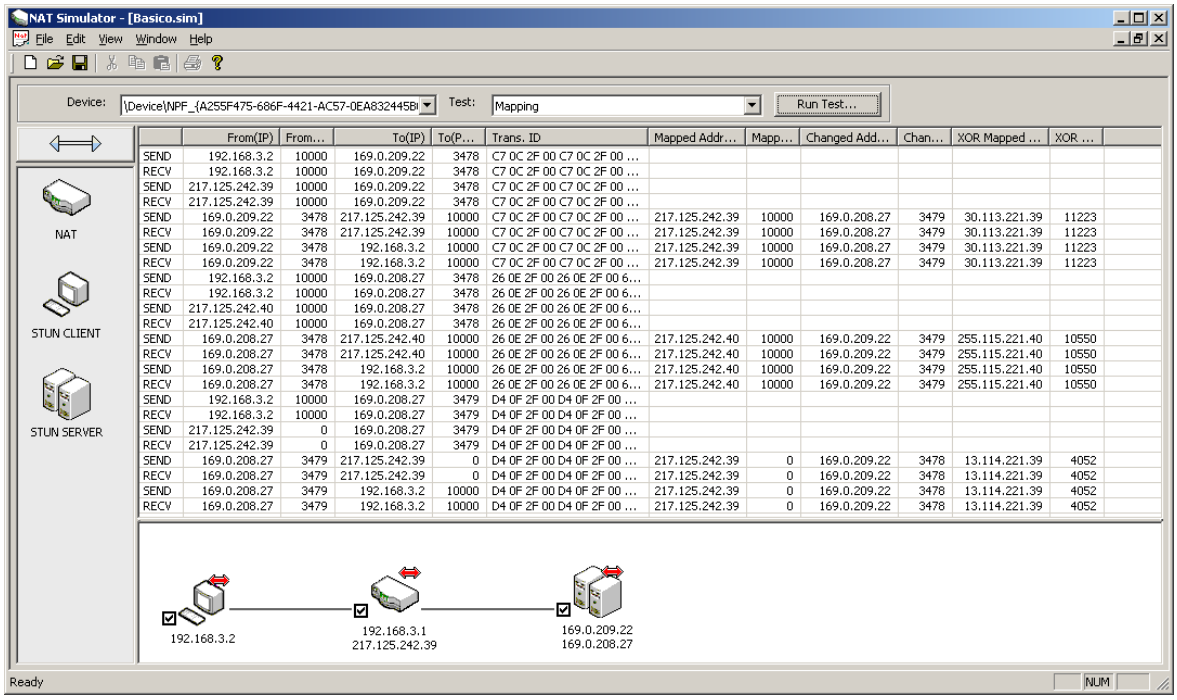

Figura 59. Validación del tipo de mapeo – Dependiente de la dirección y del puerto

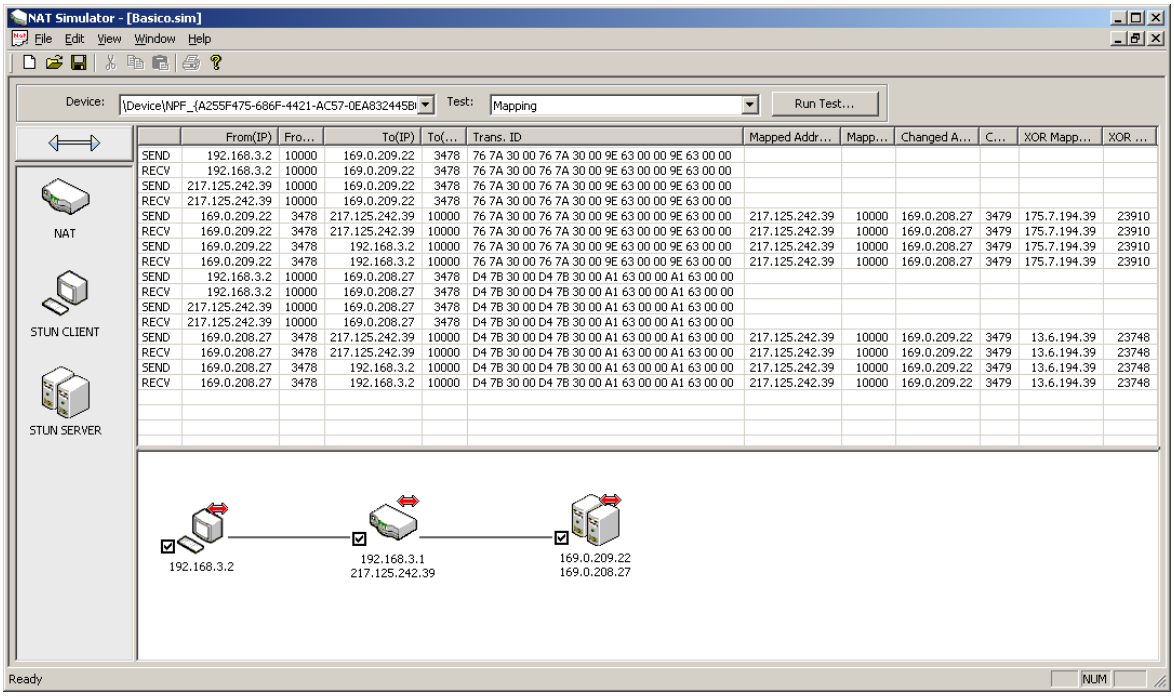

Figura 60. Validación del tipo de mapeo – Independiente

5.3.7 Un escenario de simulación más complejo

Dada la arquitectura subyacente en las librerías desarrolladas como núcleo del proyecto, es posible construir escenarios más complejos como el que se muestra en este apartado. Es importante tener en cuenta que la herramienta de creación de escenarios es solo una forma de mostrar de una manera gráfica el verdadero objetivo del proyecto. Esta herramienta permite crear escenarios que podrían dar lugar a situaciones incontroladas, como puede ser la creación de caminos alternativos, o bucles que no está contemplada pues no existe tabla de rutas alguna, sino únicamente filtros de captura.

La creación de tablas y algoritmos de enrutamiento podría ser una de las líneas de trabajo futuras en este simulador de NAT.

Como ejemplo de escenario complejo que es bien interpretado se muestra el siguiente gráfico y la ruta seguida por uno de los mensajes enviados desde el cliente hacia un servidor:

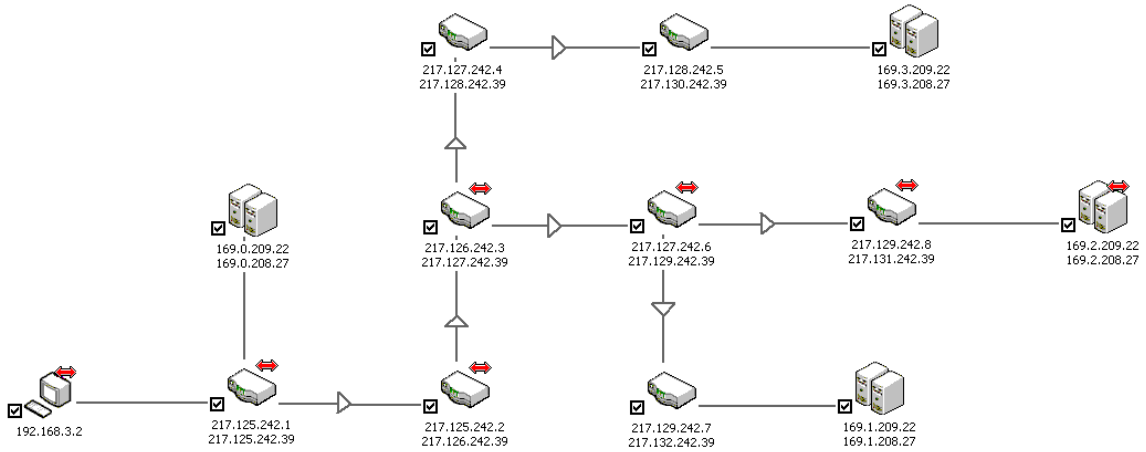

Figura 61. NAT Simulator. Escenario complejo.

	From(IP)	From...	To(IP)	To(P...	Trans. ID	Mapped Addr...	Mapp...	Changed Add...	Chan...	XOR Mapped ...	XOR ...
SEND	192.168.3.2	10000	169.2.209.22	3478	ED 35 FA 02 ED 35 FA 02 ...						
RECV	192.168.3.2	10000	169.2.209.22	3478	ED 35 FA 02 ED 35 FA 02 ...						
SEND	217.125.242.39	10000	169.2.209.22	3478	ED 35 FA 02 ED 35 FA 02 ...						
RECV	217.125.242.39	10000	169.2.209.22	3478	ED 35 FA 02 ED 35 FA 02 ...						
SEND	217.126.242.39	10000	169.2.209.22	3478	ED 35 FA 02 ED 35 FA 02 ...						
RECV	217.126.242.39	10000	169.2.209.22	3478	ED 35 FA 02 ED 35 FA 02 ...						
SEND	217.127.242.39	10000	169.2.209.22	3478	ED 35 FA 02 ED 35 FA 02 ...						
RECV	217.127.242.39	10000	169.2.209.22	3478	ED 35 FA 02 ED 35 FA 02 ...						
SEND	217.129.242.39	10000	169.2.209.22	3478	ED 35 FA 02 ED 35 FA 02 ...						
RECV	217.129.242.39	10000	169.2.209.22	3478	ED 35 FA 02 ED 35 FA 02 ...						
SEND	217.131.242.39	10000	169.2.209.22	3478	ED 35 FA 02 ED 35 FA 02 ...						
RECV	217.131.242.39	10000	169.2.209.22	3478	ED 35 FA 02 ED 35 FA 02 ...						
SEND	169.2.209.22	3478	217.131.242.39	10000	ED 35 FA 02 ED 35 FA 02 ...	217.131.242.39	10000	169.2.208.27	3479	52.182.8.37	4861
RECV	169.2.209.22	3478	217.131.242.39	10000	ED 35 FA 02 ED 35 FA 02 ...	217.131.242.39	10000	169.2.208.27	3479	52.182.8.37	4861
SEND	169.2.209.22	3478	217.129.242.39	10000	ED 35 FA 02 ED 35 FA 02 ...	217.131.242.39	10000	169.2.208.27	3479	52.182.8.37	4861
RECV	169.2.209.22	3478	217.129.242.39	10000	ED 35 FA 02 ED 35 FA 02 ...	217.131.242.39	10000	169.2.208.27	3479	52.182.8.37	4861
SEND	169.2.209.22	3478	217.127.242.39	10000	ED 35 FA 02 ED 35 FA 02 ...	217.131.242.39	10000	169.2.208.27	3479	52.182.8.37	4861
RECV	169.2.209.22	3478	217.127.242.39	10000	ED 35 FA 02 ED 35 FA 02 ...	217.131.242.39	10000	169.2.208.27	3479	52.182.8.37	4861
SEND	169.2.209.22	3478	217.126.242.39	10000	ED 35 FA 02 ED 35 FA 02 ...	217.131.242.39	10000	169.2.208.27	3479	52.182.8.37	4861
RECV	169.2.209.22	3478	217.126.242.39	10000	ED 35 FA 02 ED 35 FA 02 ...	217.131.242.39	10000	169.2.208.27	3479	52.182.8.37	4861
SEND	169.2.209.22	3478	217.125.242.39	10000	ED 35 FA 02 ED 35 FA 02 ...	217.131.242.39	10000	169.2.208.27	3479	52.182.8.37	4861
RECV	169.2.209.22	3478	217.125.242.39	10000	ED 35 FA 02 ED 35 FA 02 ...	217.131.242.39	10000	169.2.208.27	3479	52.182.8.37	4861
SEND	169.2.209.22	3478	192.168.3.2	10000	ED 35 FA 02 ED 35 FA 02 ...	217.131.242.39	10000	169.2.208.27	3479	52.182.8.37	4861
RECV	169.2.209.22	3478	192.168.3.2	10000	ED 35 FA 02 ED 35 FA 02 ...	217.131.242.39	10000	169.2.208.27	3479	52.182.8.37	4861

Figura 62. NAT Simulator. Mensajes en un escenario complejo.

5.3.8 Un escenario de ejecución de test en modo real

Podemos utilizar la herramienta para realizar pruebas en escenarios reales de comunicación a través de NAT. En este caso solo necesitaremos crear un escenario con dos elementos como el que se muestra en la figura. Lógicamente, solo podremos ejecutar aquellos test que utilicen un solo cliente debido a que la elección del interface de red se selecciona a nivel global para todos los componentes del escenario. Sería relativamente sencillo adaptar la herramienta para que cada componente usara un interface distinto si disponemos de una maquina con varios interfaces.

En la figura se observa una sesión de descubrimiento de mapeo que resulta en un NAT dependiente de la dirección y del puerto. El router usado es un Zyxel modelo P660HW-61.

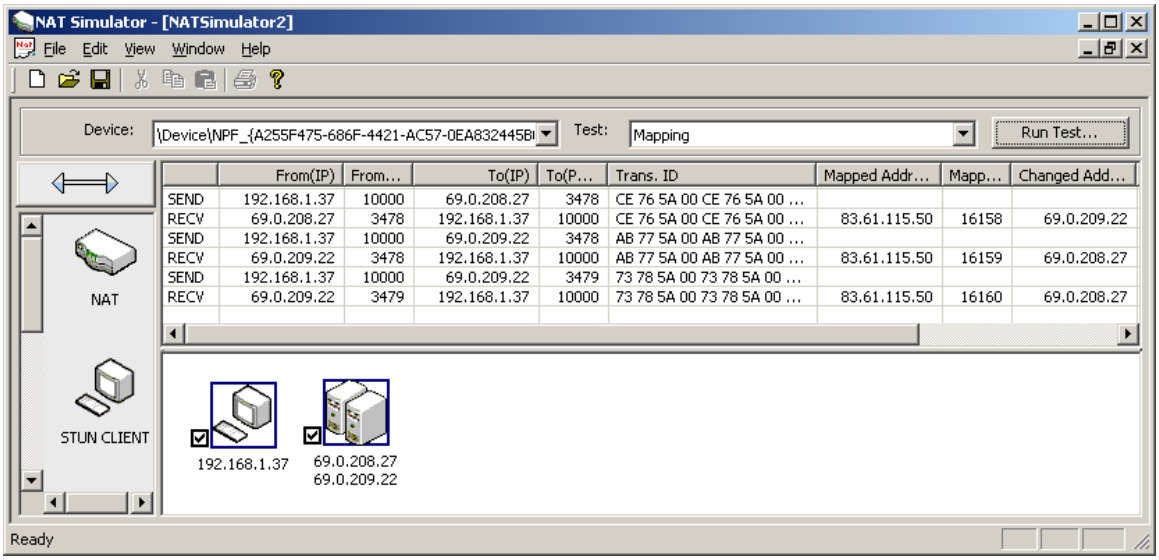

Figura 63. NAT Simulator. Mensajes en un escenario real.

5.3.9 Un escenario mixto

Finalmente, mostraremos un escenario un tanto singular. En este escenario se consigue utilizar un cliente y un NAT, ambos simulados en la conexión con un servidor real. Para lograr la conexión se asigna al NAT simulado la dirección pública del interface real de la máquina. El escenario permite crear una cascada de NAT, en la que el NAT interno es simulado y el externo es el verdadero NAT incluido en el router de acceso a Internet.

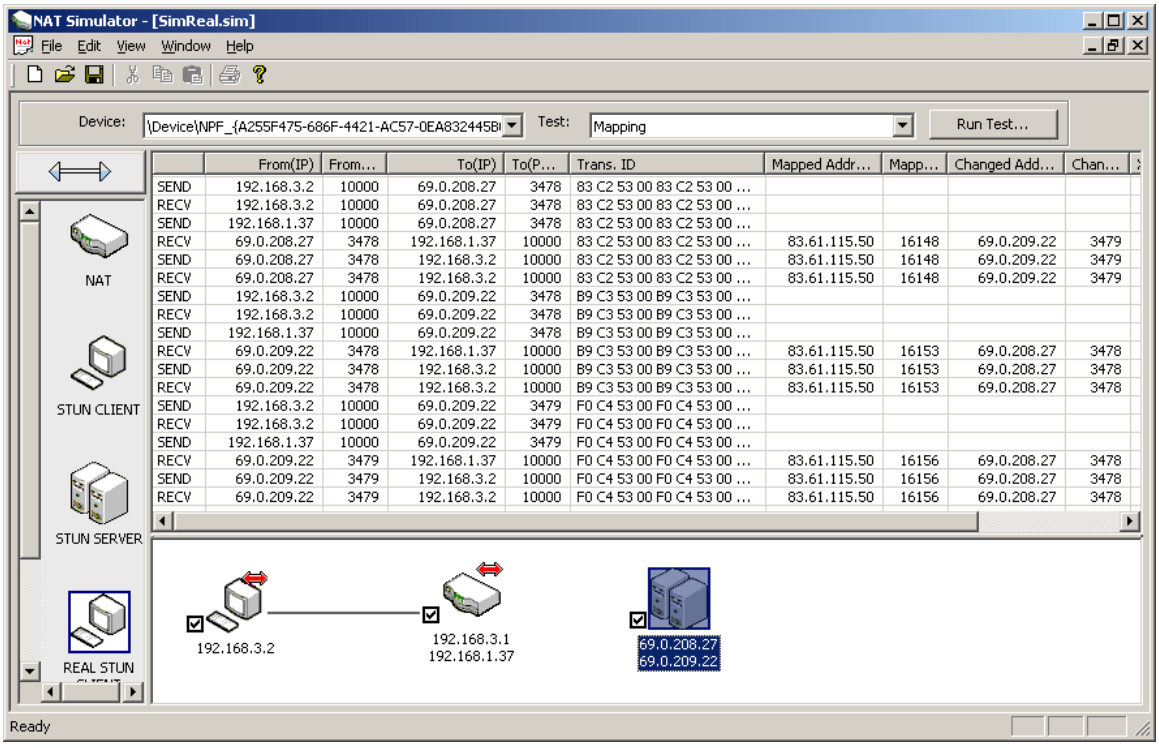

Figura 64. NAT Simulator. Mensajes en un escenario mixto.

Capítulo 6

Presupuesto

6.1 Introducción

La ejecución completa del proyecto consta de cinco fases en las que se obtienen los siguientes resultados:

Fase 1: estudio de la técnica NAT y sus implicaciones. Basándonos en los estudios realizados por el grupo de trabajo behave del IETF, obtenemos un conjunto de funcionamientos del que se extraen los requisitos que deben cumplir los fabricantes de NAT para asegurar el funcionamiento del mayor número posible de aplicaciones.

Fase 2: implementación de STUN. En esta fase desarrollamos una implementación del protocolo STUN, concretamente un cliente y un servidor STUN ligeros y configurables, que nos permiten la creación en una fase posterior de una batería de pruebas de detección basadas en los requisitos obtenidos en la primea fase.

Fase 3: desarrollo de un marco de trabajo que contenga los componentes software necesarios para la creación de la batería de pruebas de detección de NAT. En esta fase se obtendrá un conjunto de clases base con dos objetivos principales. El primero es el apoyo a la creación de las pruebas de detección y el segundo es el apoyo a la creación de escenarios de simulación de funcionamiento NAT. Principalmente se construirán las

clases destinadas a la creación de hilos de ejecución para la simulación de máquinas remotas, clases de envío y recepción de mensajes IP con cabeceras arbitrarias y un simulador de NAT configurable.

Fase 4: diseño e implementación de pruebas de detección. Basándonos en los componentes obtenidos en las fases anteriores se implementa un conjunto de pruebas suficientemente representativo de entre las obtenidas en la fase 1. Estas pruebas se podrán ejecutar primero en un entorno real y posteriormente en el entorno de simulación desarrollado en las fases 2 y 3.

Fase 5: creación de la interfaz gráfica de usuario (GUI). Para que la herramienta pueda ser utilizada como ayuda a la comprensión de la problemática del uso del NAT en escenarios reales, se implementará un interfaz de usuario que permitirá la creación y ejecución en el entorno de simulación de las pruebas desarrolladas en la fase 4.

El siguiente diagrama de Gantt muestra la evolución temporal de las fases a desarrollar. En esta se estima la duración total del proyecto en **cinco** meses.

ID	Tarea	Inicio	Fin	Duración
1	Fase 1- Estudio NAT	01/01/2009	30/01/2009	22d
2	Fase 2 - STUN Básico. Análisis	02/02/2009	04/02/2009	3d
3	Fase 2 - STUN Básico. Diseño	05/02/2009	11/02/2009	5d
4	Fase 2 - STUN Básico. Desarrollo	12/02/2009	23/02/2009	8d
5	Fase 2 - STUN Básico. Pruebas	24/02/2009	25/02/2009	2d
6	Fase 2 - STUN Básico. Documentación	26/02/2009	27/02/2009	2d
7	Fase 3 - Núcleo. Análisis	02/03/2009	05/03/2009	4d
8	Fase 3 - Núcleo. Diseño	06/03/2009	12/03/2009	5d
9	Fase 3 - Núcleo. Desarrollo	13/03/2009	26/03/2009	10d
10	Fase 3 - Núcleo. Pruebas	27/03/2009	30/03/2009	2d
11	Fase 3 - Núcleo. Documentación	31/03/2009	01/04/2009	2d
12	Fase 4 - Detección NAT. Análisis	02/04/2009	08/04/2009	5d
13	Fase 4 - Detección NAT. Diseño	09/04/2009	15/04/2009	5d
14	Fase 4 - Detección NAT. Desarrollo	16/04/2009	27/04/2009	8d
15	Fase 4 - Detección NAT. Pruebas	28/04/2009	29/04/2009	2d
16	Fase 4 - Detección NAT. Documentación	30/04/2009	01/05/2009	2d
17	Fase 5 GUI. Análisis	04/05/2009	08/05/2009	5d
18	Fase 5 GUI. Diseño	11/05/2009	15/05/2009	5d
19	Fase 5 GUI. Desarrollo	18/05/2009	27/05/2009	8d
20	Fase 5 GUI. Pruebas	28/05/2009	29/05/2009	2d
21	Fase 5 GUI. Documentación	01/06/2009	02/06/2009	2d

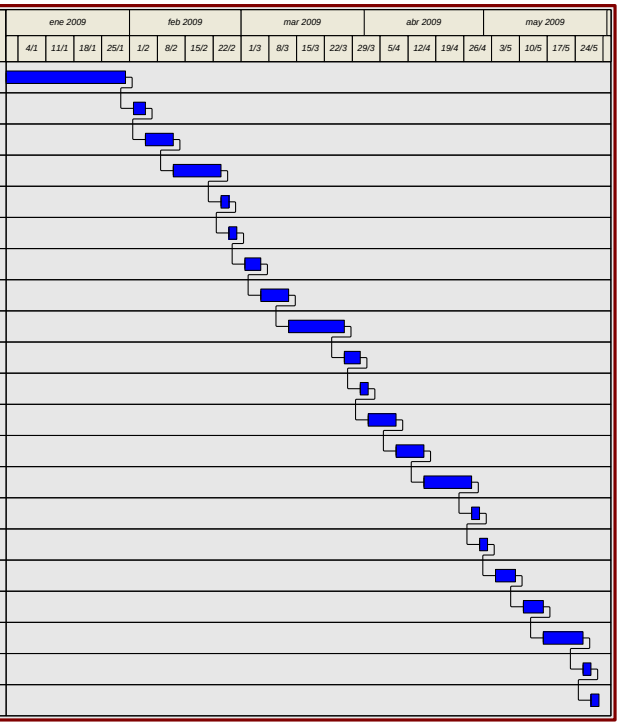

Tabla 9. Diagrama de Gantt.

6.2 Coste de material

Para la realización del proyecto se han empleado los siguientes elementos hardware y software:

HARDWARE	COSTE (€)
MacBook Pro (*)	1.800,00
Disco duro externo de 100GB	90.00
TOTAL	**1890,00**

Tabla 10. Coste de hardware.

(*) Intel Core 2 Duo, 2G RAM, 160G HD.

SOFTWARE	COSTE (€)
Windows 2000 Professional	100,00
Windows 2003 Server	700,00
Microsoft Visual Studio .NET Pro 2003	400,00
Mac OS X (*)	0,00
XCode (*)	0,00
Parallels Desktop	50,00
Virtual Box	0,00
Microsoft Office Pro 2007	709,00
Microsoft Visio Pro 2007	726,00
WinPCap 4.0.2	0,00
PThreads-Win32	0,00
Wireshark	0,00
TOTAL	**2.685,00**

Tabla 11. Coste de software.

(*) Incluido en el precio de MackBook Pro

Como valor residual de cada elemento hardware y software se ha considerado el 15% de su valor de adquisición, y se ha calculado una amortización prolongada durante cinco años (dichos elementos sólo se han empleado en el proyecto durante los cinco meses que ha durado).

MATERIAL	COSTE (€)	COSTE EFECTIVO (€)
Hardware	1.890,00	133,88
Software	2.685,00	190,19
TOTAL MATERIAL	**4.575,00**	**324,06**

Tabla 12. Gastos efectivos en hardware y software.

Además se han incurrido los siguientes gastos fijos:

GASTOS FIJOS	COSTE (€)
Puesto de trabajo (*)	1.500,00
Conexión ADSL (**)	300,00
Desplazamientos (***)	27,00
Material de oficina	500,00
TOTAL GASTOS FIJOS	**2.327,00**

Tabla 13. Gastos fijos.

(*) Coste del lugar de trabajo por mes 300 € durante 5 meses.
(**) Coste de conexión mensual de 60 € durante 5 meses.

(***) Coste por Km. de desplazamiento de 0,3 € por 15 visitas por 20 Km por visita.

Por lo tanto, la suma total de los costes de hardware y software más los gastos fijos son:

MATERIAL	COSTE (€)
Hardware y Software	324,06
Gastos fijos	2.327,00
TOTAL	**2.651,06**

Tabla 14. Costes de material.

6.3 Coste de personal

Para el desarrollo de este proyecto se han invertido cinco meses efectivos, con un total de 109 días laborables y una dedicación media de 8 horas diarias. Esto supone un total de 872 horas efectivas dedicadas a la realización del presente proyecto.

El precio hora para el desarrollo del proyecto se ha dividido en dos bloques de facturación correspondientes al estudio preliminar, análisis y diseño por una parte y al desarrollo, pruebas y documentación por otra. Para el primer bloque el precio/hora es de 70 € mientras que para el segundo bloque el precio/hora es de 50 €.

Para la labor de estudio preliminar, análisis y diseño se han dedicado 59 días a razón de 8 horas/día lo que supone un total de 472 horas y el presupuesto es de 33.040 €.

Para el desarrollo, pruebas y documentación se han invertido 50 días a razón de 8 horas/día, lo que supone un total de 400 horas y el presupuesto es de 20.000 €.

La dirección del proyecto ha sido llevada a cabo por el Dr. D. Francisco Valera Pintor, que ha empleado aproximadamente unas 50 horas en llevar a cabo esta tarea, lo que supone un coste de 3.500 €.

Los costes de personal totales se muestran en la siguiente tabla:

PERSONAL	COSTE (€)
Estudio preliminar, análisis y diseño	33.040,00
Desarrollo, pruebas y documentación	20.000,00
Dirección del proyecto	3.500,00
TOTAL	**56.540,00**

Tabla 15. Costes de personal.

6.4 Coste total

En el cálculo del coste total se tienen en cuenta las estimaciones realizadas tanto para los costes de material como para los costes de personal. A la cantidad resultante hay que aplicarle el incremento correspondiente al IVA, que se establece en un 16%. El desglose se muestra en la siguiente tabla:

CONCEPTO	**COSTE (€)**
Material	2.651,06
Personal	56.540,00
Base imponible	59.191,06
IVA (16%)	9.470,57
TOTAL	**68.661,63**

Tabla 16. Coste total del proyecto.

El presupuesto total de este proyecto asciende a la cantidad de SESENTA Y OCHO MIL SEISCIENTOS SESENTA Y UN EUROS CON SESENTA Y TRES CÉNTIMOS.

Leganés a 1 de Octubre de 2009

El ingeniero proyectista

Fdo. Francisco José Blázquez Sánchez

Capítulo 7

Conclusiones y trabajos futuros

7.1 Introducción

En este capitulo se comentan las conclusiones finales y se presentan los posibles trabajos futuros que se podrían abordar a partir de los resultados obtenidos en la elaboración del proyecto.

7.2 Conclusiones

El proyecto se ha abordado en dos grandes bloques independientes. El primer bloque se ha centrado en el estudió de la problemática de NAT y la creación de herramientas software que permitan la observación dinámica de su funcionamiento. El segundo bloque se ha centrado en la elaboración de una plataforma de pruebas que permite aislar el problema y estudiarlo desde una perspectiva más sencilla en cuanto a los elementos hardware necesarios.

Gracias a esta división inicial, podemos extraer conclusiones en dos sentidos diferentes, como son las derivadas del funcionamiento del NAT y las posibilidades y

carencias que ofrece, así como las posibles soluciones a estas deficiencias, y las obtenidas tras el desarrollo de la herramienta de simulación y sus posibles usos futuros en otros estudios de conexión basados en TCP.

Como ya se ha comentado en este documento, NAT ha sido una técnica que se ha extendido en la red de forma un tanto incontrolada, principalmente debido a la sencillez de su implantación en los extremos de la comunicación y también a los buenos resultados que ha ofrecido en multitud de situaciones de escasez de direcciones IP, abaratando la conexión a Internet y permitiendo de esta forma el acceso a muchas personas que de otro modo hubieran tenido dificultades para disfrutar de este servicio. Podemos decir, siendo un poco osados, que NAT ha participado de forma directa en el incremento espectacular que ha experimentado el uso de Internet como herramienta de trabajo y de ocio en la última década.

Este crecimiento de Internet ha evolucionado en el sentido de la participación de los usuarios finales, de forma que estos han pasado de ser elementos un tanto pasivos como consumidores de información que acceden a grandes servidores a verdaderos participantes activos en el intercambio de información y en el que cada uno de los extremos es en si mismo, productor y consumidor de una información que se encuentra segmentada y distribuida y que por lo tanto ha de ser visible y alcanzable por el resto de usuarios. En estos nuevos escenarios NAT supone un verdadero problema y parece difícilmente superable en la actualidad.

Por lo tanto se puede decir que NAT ha sido un acierto en cuanto que ha cumplido con la misión de acercar a la red a un mayor número de individuos que los inicialmente previstos, pero que ha fallado en la falta de previsión de posibles problemas futuros.

Por otra parte resulta difícil calibrar el coste que supone la actualización de NAT para salvar en la medida de lo posible las dificultades detectadas. Es también muy arriesgado perseverar en el uso de NAT como solución definitiva a la escasez de direcciones IP. En cualquier caso, todas las mejoras que se puedan introducir en el NAT actual son muy interesantes, pero en ningún caso se debería considerar como una alternativa a la implantación de IPv6 puesto que esta es un concepto mucho más amplio que abarca otros campos de mejora además de la escasez de direcciones IP.

Bajo mi punto de vista y basándome en los resultados obtenidos en los test implementados, pienso que es imprescindible que el funcionamiento de NAT sea homogéneo y determinístico. Homogéneo en cuanto que no deberían existir diferencias de funcionamiento bajo condiciones de trafico idénticas y determinístico en cuanto que se debería conocer a priori el comportamiento del NAT ante cualquier situación. Estas características facilitarían el desarrollo de protocolos más robustos y seguros. Por otra parte, los estudios realizados por el IETF ponen de manifiesto una serie de funcionalidades o comportamientos del NAT que no permiten o terminan la conexión en determinadas situaciones de carga. En estos casos, el NAT debe ofrecer alternativas de conexión con las que los protocolos puedan lidiar para mantener la conexión activa.

Por otra parte el uso de NAT ha puesto en evidencia algunos detalles de implementación de ciertos protocolos que hacen suposiciones en cuanto a la numeración de los puertos utilizados. Es por lo tanto necesario un esfuerzo adicional por parte de los desarrolladores de protocolos para la convivencia de sus aplicaciones con NAT.

En cuanto a las posibilidades de usos futuros del simulador desarrollado, podemos concluir que la técnica utilizada posibilita tanto el crecimiento de la herramienta como marco de pruebas para la detección de NAT como la utilización de las librerías de base implementadas en otros escenarios de conexión vía TCP.

La arquitectura creada para esta simulación, basada en la ejecución de hilos independientes y usando la captura de tráfico a nivel IP, es muy apropiada para la ejecución de cualquier escenario de comunicaciones sobre IP, ya sea a nivel de red o a nivel de aplicación.

7.3 Trabajos futuros

El núcleo de la herramienta ha sido desarrollada en C++, usando librerías estándar y portables a entornos Windows, Linux y Mac. La elección de este lenguaje de programación permite la portabilidad de la aplicación a cualquiera de estos sistemas operativos. Como a se ha comentado en este documento, la utilidad gráfica para la creación de escenarios ha sido desarrollado en C++ utilizando la librería MFC de Microsoft y es por lo tanto poco portable a otras plataformas distintas de Windows. Esta circunstancia, unida a la decisión de los desarrolladores de Windows de limitar el uso de sockets raw en sus sistemas operativos, concretamente en Windows XP SP 2 y Windows Vista, hace pensar que la elección de esta librería para el desarrollo de la utilidad gráfica no es muy acertada. Inicialmente no estaba previsto el desarrollo de dicha utilidad, pero dado que el desarrollo de la misma ayuda en gran medida a la comprensión de los resultados de los test implementados, se decidió desarrollar con el menor coste posible y por ello se decidió la utilización de la librería MFC.

Uno de los primeros posibles trabajos futuros consistiría en la migración de esta herramienta grafica a otras librerías mas portables como podría ser QT, que se encuentra disponible en versiones de código libre para varias plataformas, incluyendo Windows, Linux y Mac.

En cuanto al estudio de NAT, se pueden seguir implementando otros test que no se han desarrollado en este proyecto, como son la el funcionamiento en horquilla o la fragmentación de paquetes de entrada o salida. También es posible actuar a nivel de paquetes ICMP. En este sentido hay otras posibilidades de expansión del simulador en el nivel de enrutamiento de paquetes, dotando a la herramienta de una verdadera tabla de rutas para cada elemento de simulación.

Otra posible vía futura consiste en la adaptación de la herramienta a IPv6 puesto que la problemática NAT es muy parecida a otros escenarios de conexión a través de dispositivos que actúan como puntos de entrada en una red privada y que previsiblemente seguirán siendo utilizados en escenarios con IPv6 ya implantado.

Por último, dado que en breve se extenderá el uso de los nuevos sistemas operativos de 64 bits, sería interesante la migración del código a estos nuevos sistemas.

Glosario

ADSL	*Asymmetric Digital Subscriber Line*
ALG	*Application Layer Gateway*
API	*Application Program Interface*
BEHAVE	*Behavior Engineering for Hindrance Avoidance*
DHCP	*Dynamic Host Configuration Protocol*
DNAT	*Destination Network Address Translation*
DoS	*Denial Of Service*
ICE	*Interactive Connectivity Establishment*
ICMP	*Internet Control Message Protocol*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
IPv4	*Internet Protocol Version 4*
IPv6	*Internet Protocol Version 6*
ISP	*Internet Service Provider*
LAN	*Local Area Network*
MFC	*Microsoft Foundation Classes*
MMUSIC	*Multiparty Multimedia Session Control*
MTU	*Maximum Transmission Unit*
NAT	*Network Address Translation*
PAT	*Port Address Translation*
RFC	*Request For Comments*
RTCP	*Real Time Control Protocol*
RTO	*Retransmission Time Out*
RTP	*Real Time Transport Protocol*
SIP	*Session Initiation Protocol*
STUN	*Session Traversal Utilities for NAT*
TCP	*Transmisión Control Protocol*
TLV	*Type Length Value*
UDP	*User Datagram Protocol*
UNSAF	*UNilateral Self-Address Fixing*
WAN	*Wide Area Network*

Referencias

[RFC 0768] *User Datagram Protocol*
http://tools.ietf.org/html/rfc0768

[RFC 1631] *The IP Network Address Translator (NAT)*
http://tools.ietf.org/html/rfc1631

[RFC 1918] Address Allocation for Private Internets
http://tools.ietf.org/html/rfc1918

[RFC 2663] IP Network Address Translator (NAT) Terminology and Considerations
http://tools.ietf.org/html/rfc2663

[RFC 2993] Architectural Implications of NAT
http://tools.ietf.org/html/rfc2993

[RFC 3489] STUN - Simple Traversal of User Datagram Protocol (UDP)
Through Network Address Translators (NATs)
http://tools.ietf.org/html/rfc3489

[RFC 3550] RTP: A Transport Protocol for Real-Time Applications
http://tools.ietf.org/html/rfc3550

[RFC 3605] Real Time Control Protocol (RTCP) attribute in
Session Description Protocol (SDP)
http://tools.ietf.org/html/rfc3605

[RFC 4787] Network Address Translation (NAT) Behavioral Requirements
for Unicast UDP

http://tools.ietf.org/html/rfc4787

[POSIX] Portable Operating System Interface
http://standards.ieee.org/regauth/posix/

[MSDN] Microsoft Developer Network
http://msdn.microsoft.com